# CompleteFTP

# User's Guide

# Table of Contents


- Overview
  - Requirements
  - How to select the edition
  - License Agreement
  - Revision History
- Installation
  - Uninstalling
  - How to run the server
  - How to use the manager
  - How to migrate to another machine
  - How to automate installation
  - How to activate an installation of CompleteFTP
  - How to upgrade
- Getting started tutorials
  - Connecting with Filezilla
  - Connecting with WinSCP
- Overview panel
- Site configuration
  - FTP/FTPS settings
  - SFTP/SCP/SSH settings
  - HTTP/HTTPS settings
  - File Sharing Settings
  - File System Settings
  - General User Settings
  - IP Filtering and Auto-Banning
  - Limits and Timeouts
  - Messages
  - How to secure your server
  - How to set a default Windows domain
  - How to use HTTP/HTTPS
  - How to use the File Manager
  - How to format HTML listings
  - How to customize the welcome page
  - How to use CompleteFTP with firewalls
  - How to set an external IP address
  - How to set a PASV port range
  - How to set up the server certificate
  - How to set up a CA certificate
  - How to generate a CSR
  - How to purchase a CA certificate
  - How to install a CA certificate
  - How to import SSL certificates in other formats
  - How to set up the server SSH keys
  - How to use SSH terminal access
  - How to use SSH tunneling
  - How to configure anonymous logins
  - How to use multiple sites
- File System
  - How to add a virtual folder
  - How to add a Windows folder
  - How to add a Windows special folder
  - How to add a Network or Macro folder
  - How to add a Gateway folder
  - How to insert macros into folder-names
  - How to give every user the same directory structure
  - How to set a folder's permissions
  - How to extend the filesystem
  - How to stop executables being uploaded
  - How to mount ZIP files as directories
  - How to enable backslash separators
- Users
  - General User Settings
  - How to add a user
  - How to add a Windows user
  - How to add a non-Windows user

- How to add multiple users
- How to change user properties
- How to configure database users
- How to configure automatic Windows users
- How to configure gateway users
- How to set user quotas
- How to set up public key authentication (SFTP/SCP)
- How to control user access by IP address
- How to set up client certificates (FTPS)
- How to make a user's home dir root
- How to set password policies

File sharing
- How to enable file sharing
- How to install CompleteBox
- File sharing vs file storing
- How to share files with CompleteBox
- How to store files with CompleteBox
- File synchronization
- How to configure CompleteBox

Clustering
- How to enable clustering
- How to change primaries
- How to resolve problems
- How to activate cluster nodes
- How to upgrade a cluster

Multiprotocol Gateway
- How to configure a gateway
- How to configure a gateway authenticator

Groups

Events
- How to set up email notifications
- How to set up process triggers
- Scheduled events
- How to write FTP scripts
- FTP script command reference
- Event macros
- How to write your own events

Web-Apps
- Introduction to JSS
- JSS Features
- JSS Templates
- JSS Examples
- JSS API Guide
- JSS RPC Reference

Custom Extensions
- How to write an extension
- Authentication extensions
- How to write a simple authentication extension
- How to write an advanced authentication extension
- File-system extensions
- Web-app extensions
- Custom command extensions
- Event extensions
- JSS extensions
- Custom command scripting
- How to add logging

Monitoring
- How to view the server log files
- How to view the manager log files
- How to change the log file location
- How to enable auditing
- How to modify the logging configuration
- How to produce clean log files
- How to get diagnostics for support

Performance

Administrator Access
- How to change the administrator site settings
- How to add extra administrator users

- How to do command-line administration
- How to do script-based administration
- Command-line reference

Security
- How to disable SSL 3.0

Licensing
- How to migrate to another machine
- How to resolve restricted mode
- How to automatically import a license

Support
- How to contact suppport
- FAQ
- CompleteBox revision history

FTP Protocol Overview
- Active and Passive modes
- FTP Commands
- Sample Scenarios
- Data types
- Session commands
- File commands
- Directory commands

FTPS - Securing FTP with TLS
- Implicit FTPS and Explicit FTPS
- Securing Control and Data Channels
- FTPS Commands
- FTPS Usage

The Essentials of FTP Security
- Public Key Cryptography
- Certificates and Certificate Authorities (CAs)
- Obtaining Keys and Certificates
- Server and Client Validation
- Hostname Checking
- Selecting Ciphers

SFTP and SCP
- SSH - Secure Shell
- SCP - Secure Copy
- SFTP - SSH File Transfer Protocol
- Comparison of FTPS and SFTP

HTTP Protocol Overview
- HTTP methods
- HTTP sessions
- HTTPS
- About this manual

# Product Overview

**CompleteFTP** is a high performance Windows file server supporting secure file transfer via FTPS (FTP over SSL), SFTP (FTP over SSH), HTTP, HTTPS, SCP and SSH. CompleteFTP also supports web applications written in Javascript, and collaborative file sharing.

Easy to install, fast, and easy to configure, CompleteFTP is the ideal server solution for securing file transfers on Windows. Features include:

- Full support for FTP and FTPS (FTP over SSL). Includes implicit and explicit FTPS.
- Full support for SFTP (FTP over SSH).
- Support for HTTP and HTTPS (Professional and Enterprise Editions), including support for Javascript FileManager, directory listings and HTTP uploads (unlimited in size!). Custom MIME types are supported.
- Full support for SCP (Professional and Enterprise Editions).
- Support for SSH tunneling/SSH local port forwarding (Professional and Enterprise Editions).
- Support for SSH terminals (Professional and Enterprise Editions).
- Support for collaborative file sharing (Professional and Enterprise Editions).
- Support for web applications using server side Javascript (JSS) (Professional and Enterprise Editions).
- Support for customized and automated installation (Enterprise Edition only).
- Support for customised authentication, file system, event and custom command extensions and scripting (Enterprise Edition only).
- Support for generation of Certificate Signing Requests (CSRs) for submission to a Certificate Authority (CA) to obtain a CA certificate.
- Ability to act as a multiprotocol gateway for other servers. Use CompleteFTP to provide secure access via FTPS or SFTP to an existing FTP server. Add FTP or FTPS support to an existing SFTP server (Enterprise Edition only).
- Support for file sizes > 4 GB.
- User disk and bandwidth quotas.
- Extensive real-time charts for performance monitoring. (Professional and Enterprise Editions).
- Email notification of events such as upload, download, delete (Professional and Enterprise Editions).
- Process Triggers for launching server processes, Powershell scripts, batch scripts or FTP scripts when events such as uploading or downloading occur (Professional and Enterprise Editions).
- IP filtering and auto-banning.
- Remote administration.
- Virtual File System.
- ZIP-file navigation. Users can change into folders within ZIP files and download specific files without downloading the entire ZIP file.
- Windows and non-Windows users. Automatic Windows users feature. Support for Windows domain users, and can set a default domain.
- Support for external users (e.g. stored in an external database).
- Unlimited users, fast performance.
- SSL self-signed certificate generator.

## Requirements

**CompleteFTP** will run on all versions of Windows, including Windows XP, Windows Vista, Windows 7, Windows 8, Windows Server 2003, Windows Server 2008 and Windows Server 2012. Both 32 bit and 64 bit versions of Windows are supported. CompleteFTP also runs on Windows Azure and Amazon's Elastic Compute Cloud (EC2).

In terms of processing power, we recommend at least a dual core CPU, as encrypted transfers are CPU intensive. If the network connection is fast, CPU will be the bottleneck. If CompleteFTP is being run in a VM, we recommend allocating at least 2 vCPUs to the VM. CompleteFTP has a low memory footprint so memory should never be an issue, although in a VM be sure to allocate at least 1 GB.

# How to select the edition

**CompleteFTP** comes in three editions, the *Standard Edition*, the *Professional Edition* and the *Enterprise Edition*.

The *Standard Edition* is for those who want the essential features of FTP, FTPS and SFTP for the lowest possible price.

The *Professional Edition* supports all standard protocols such as FTP, FTPS, SFTP, SCP, HTTP & HTTPS. It supports Windows domain users, event notifications and process triggers, quotas, SSH terminal access and advanced file permissions. It also supports the Javascript FileManager if HTTP is enabled, filesharing and web-apps (server-side Javascript). For many users, the Professional Edition is the appropriate choice.

The *Enterprise Edition* should be considered by those who want to add custom functionality of their own to CompleteFTP, who require clustering support, or who need multiple sites supported on the one server. The Enterprise Edition also supports the multi-protocol gateway, which allows CompleteFTP to add secure protocols to other servers, as well as the Javascript FileManager.

Note that the trial installation can easily be upgraded from a lower edition to a higher one from the *Licensing* tab, so that if it was initially installed as Professional and you wish to evaluate the Enterprise Edition, you can upgrade with a mouse click. Installations cannot be downgraded to a lower edition.

The differences between the editions are shown in a table here.

Note that CompleteFTP is licensed on a per machine basis, i.e. one license is required for each machine it is installed on, unless the *Corporate License* is purchased. The *Corporate License* permits installation on any number of machines in the corporation, and optionally provides a universal activation key so online activation is not required.

**Upgrading**

If you have purchased the Standard or Professional Edition but subsequently decide that you require the features of a higher edition, you can upgrade for the cost of the price difference plus a USD 30 administration fee - please contact us to arrange this (and include your purchase reference in your email).

# CompleteFTP Server Software License Agreement

We **Enterprise Distributed Technologies Ltd** grant you license to use this Software on the following terms:

1. **License and support**

    1. This Software is licensed and not sold to you. Your use of this Software is governed solely by the terms of this License. This License is personal to you.

    2. This Software is owned by us, save for certain elements of it which are owned by The KPD-Team; Chew Keong Tan; Routrek Networks, Inc; ComponentAce; Developer Express Inc; The Mentalis.org Team; Novell, Inc; Motus Technologies Inc; Studio 42; Evgen Noskov; and The Outercurve Foundation. This Software is protected by copyright law, and the owners reserve ownership of all Intellectual Property Rights in it, and all rights other than those expressly granted by this Agreement.

    3. You must take reasonable steps to protect the Software from unauthorised copying, publication, disclosure or distribution.

    4. Technical support is initially available via email for 12 months after the date of purchase (and can be renewed annually). We will endeavour to respond to queries as soon as possible, but cannot guarantee a response within any particular time frame. Note that if excessive use is being made of technical support, an additional payment not exceeding the payment required for 12 months' technical support will be required. Excessive use is defined on a calendar year basis as exceeding five separate issues for a Single Server License, eight separate issues for a 2 Server License, fifteen separate issues for a 5 Server License, twenty separate issues for a 10 Server License, and fifty separate issues for a Corporate Server License.

    5. Support agreements must be renewed within 6 months of expiry, otherwise the higher expired support renewal price will apply.

    6. Where use is permitted by this License, such use includes

        1. the making of no more than a reasonable number of backup copies of the Software; and

        2. the right to utilise and make prints for your own personal use (but not otherwise copy) any instructional and/or operational manuals relating to the Software.

2. **Trial License**

    Until and unless you buy a Full License for the use of the Software:

    1. Your rights are limited to

        1. use for evaluation purposes only, and

        2. use for 30 days following first installation only, thereafter, the Trial License converts to a Developer License (the 30 day Trial License period may be extended on request).

    2. You may not modify the Software. You may not disassemble, decompile, or reverse engineer the Software, or otherwise attempt to discover the source code of the Software.

3. **Developer License**

    Until and unless you buy a Full License for the use of the Software:

    1. Your rights are limited to use for evaluation and code-development purposes only, indefinitely.

    2. You may not access the Software in any way from a machine other than the machine on which the Software is installed.

    3. You may not modify the Software. You may not disassemble, decompile, or reverse engineer the Software, or otherwise attempt to discover the source code of the Software.

4. **Full License**

    Subject to payment of the full License fee:

    1. Server Application. We grant to you a non-transferable and non-exclusive license to install the Server Application for use on the number of server machines for which you have purchased separate licenses.

    2. In the case of a Corporate License, you are entitled to install the Server Application on any number of machines within your organization, i.e. the legal entity which purchased the license. A Corporate License is not a multi-national license, i.e. it is applicable only within the country of purchase.

    3. Administrator Application. We grant to you the right to install and use the administrator application on an unlimited number of machines provided that you are licensed to use the Server Application.

    4. You may not modify the Software. You may not disassemble, decompile, or reverse engineer the Software, or otherwise attempt to discover the source code of the Software.

5. **CompleteBox User Licenses**

    Subject to payment of the full license fee:

    1. We grant to you one CompleteBox User License, which permits a single CompleteBox user to connect to the Server Application for the purposes of file sharing. A CompleteBox user is a human being, not a user account shared between people. Each person using CompleteBox requires a CompleteBox User License. They may use their license from any client machine, i.e. a person with a CompleteBox User License can connect from multiple machines.

2. Subject to the payment of the CompleteBox User License fee (which is optional), we grant you the additional number of CompleteBox User Licenses that are purchased, which permits the specified number of users to connect to the Server Application for the purposes of file sharing.

6. **Warranties, and Limitation of Liability:**

    1. You acknowledge that

        1. you have had the opportunity to evaluate the Software without charge, and that you have satisfied yourself that it meets your requirements in all material respects

        2. it is not technically practicable to guarantee software to be error-free, and you agree that if any such errors are found to exist they shall not constitute a breach of this Agreement

        3. the Software is provided for use by you as an expert, and that where you decide to incorporate the Software in an application created by you, it is solely your responsibility to carry out all appropriate testing and to determine its suitability for your intended purpose. You will therefore indemnify us against any and all claims that may be brought by a third party in relation to any applications distributed by you which incorporate the Software, whether or not the terms of the License you have bought permit such distribution.

    2. THE SOFTWARE IS NOT WARRANTED TO BE FAULT-TOLERANT, AND IS NOT INTENDED FOR THE DESIGN, CONSTRUCTION, MAINTENANCE, OPERATION, CONTROL, OR ANY OTHER USE IN CONNECTION WITH HIGH RISK SYSTEMS. WE SPECIFICALLY DISCLAIM ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR PURPOSE IN CONNECTION WITH HIGH RISK SYSTEMS. 'High Risk Systems' means systems in environments requiring fail-safe performance (such as nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems), in which the failure of the software could lead directly to death, personal injury, or severe physical or environmental damage. You agree that you will not use the software for any purpose in connection with High Risk Systems.

    3. THIS SOFTWARE IS PROVIDED ``AS IS'', AND TO THE EXTENT PERMITTED BY THE APPLICABLE LAW, WE DISCLAIM ALL WARRANTIES WITH RESPECT TO THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE.

    4. ALTHOUGH WE DO NOT WARRANT THAT THE SOFTWARE SUPPLIED HEREUNDER SHALL BE FREE FROM ALL KNOWN VIRUSES WE HAVE USED COMMERCIALLY REASONABLE EFFORTS TO CHECK FOR THE MOST COMMONLY KNOWN VIRUSES. YOU ARE NEVERTHELESS SOLELY RESPONSIBLE FOR VIRUS SCANNING THE SOFTWARE.

    5. WE SHALL NOT BE LIABLE TO YOU OR TO ANYONE ELSE FOR ANY LOSS OR DAMAGE WHATSOEVER OR HOWSOEVER CAUSED ARISING DIRECTLY OR INDIRECTLY IN CONNECTION WITH THIS LICENSE, THE SOFTWARE, ITS USE OR OTHERWISE, EXCEPT TO THE EXTENT THAT SUCH LIABILITY MAY NOT BE LAWFULLY EXCLUDED UNDER THE APPLICABLE LAW.

    6. NOTWITHSTANDING THE GENERALITY OF THE ABOVE, IN NO EVENT WILL WE BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL LOSS OR DAMAGE WHICH MAY ARISE IN RESPECT OF THE SOFTWARE, ITS USE, OR IN RESPECT OF OTHER EQUIPMENT OR PROPERTY, OR FOR LOSS OF PROFIT, BUSINESS, REVENUE, GOODWILL OR ANTICIPATED SAVINGS, EVEN IF WE HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSS.

    7. IF ANY EXCLUSION OR LIMITATION CONTAINED IN THIS LICENSE SHALL BE HELD TO BE INVALID FOR ANY REASON AND WE BECOME LIABLE FOR LOSS OR DAMAGE THAT MAY LAWFULLY BE LIMITED, SUCH LIABILITY SHALL BE LIMITED TO THE LICENSE FEE PAID BY YOU FOR THE SOFTWARE.

    8. We do not exclude liability for death or personal injury to the extent only that it arises as a result of our negligence or that of our employees, agents or authorized representatives.

    9. So far as any parts of this Software which were not created by us are concerned, THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL ANY PERSON WHO HAS CONTRIBUTED TO OR IS THE OWNER OF ANY PART OF THIS SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This provision shall operate for the benefit of any and all such persons.

7. **Miscellaneous**

    1. You acknowledge that these terms supersede all prior agreements, and are complete and exclusive. No oral or written information given by us or on our behalf shall create a warranty or collateral contract, or in any way increase the scope of this warranty in any way, and you may not rely on any such advice.

    2. If any provision in this Agreement shall be determined to be invalid, such provision shall be deemed omitted; the remainder of this Agreement shall stand.

    3. This License shall be governed by the laws of Australia.

## CompleteFTP revision history

| Version | Changes |
|---|---|
| Version 8.5.0<br>(30 November 2015) | • Password hashing now uses PBKDF2-SHA256.<br>• New command-line commands for administering permissions (chmod, chown).<br>• Support for TLS_FALLBACK_SCSV in TLS.<br>• Fix bug in SSL3 renegotiation.<br>• Change anonymous FTP behaviour so that if disabled there is still a password prompt.<br>• Fix issue where login failures did not trigger the login event when set.<br>• Remove RC4 support from default ciphers (as no longer regarded as secure).<br>• Fix issue where downloading files with special chars in their name from HTTP works on Firefox but not with Internet Explorer.<br>• Fix bug where all connections in the manager application are shown as from one user. |
| Version 8.4.0<br>(9 October 2015) | • Now supports TLS 1.2.<br>• Now supports TLS secure renegotiation.<br>• Fix bug where downloading in SFTP via Gateway failed. |
| Version 8.3.3<br>(7 September 2015) | • More efficient memory utilization, and now uses .NET 4.x framework if it is installed.<br>• Bugfix for scheduled events, where some events were not firing due to a race condition.<br>• Bugfix for downloading from a zip folder.<br>• Fixed OpenSSH 6.7 (and higher) issue using HMAC SHA2 with SFTP.<br>• Fixed incorrect IP address in logging. |
| Version 8.3.2<br>(22 July 2015) | • Now reuses PASV port for the same client, meaning PASV port range is used more efficiently.<br>• Fix to FTPS implicit mode.<br>• Better throttling of server messages being sent to clients, reducing memory consumption.<br>• Fix to SSH key re-exchange when initiated by clients.<br>• New version of CompleteBox (1.1.1) included. |
| Version 8.3.1<br>(17 June 2015) | • Replace Microsoft HMAC in SSH/SFTP as it was causing problems for a user.<br>• Fix a directory traversal security vulnerability.<br>• Minor fixes to logging. |
| Version 8.3.0<br>(8 May 2015) | • Added scheduled events. This means events can be set up to fire at a predefined schedule, such as once per hour or at midnight every weekday.<br>• Selectable ADO.NET providers for database authenticator.<br>• Add SHA1 to database authentication option, so now MD5 and SHA1 are supported.<br>• A user's Active Directory home directory can now be set as a macro variable.<br>• Set service recovery options to automatically restart for new installations.<br>• If the anonymous user is disabled in FTP, give an error rather than prompting for password.<br>• Fix issue with certificate signing requests that meant Verisign/Symantec would not accept them.<br>• Make IPAddresses configuration field larger so more network interfaces can be supported.<br>• Gateway authenticator fix - for multiple authenticators, the connection details of the first remote server were being re-used. |

| | |
|---|---|
| Version 8.2.0<br>(18 March 2015) | • Per-user IP filtering.<br>• Add SHA-1 password hashing to Authenticator. Now extensions can use SHA-1 hashed passwords.<br>• Revert to using SHA-1 in certificate signing requests (CSR) because Verisign doesn't accept SHA-256 requests!<br>• In CompleteFTP manager, the FTP/FTPS Server Certificate property is now also available under Advanced HTTP/HTTPS Settings.<br>• There is now an Extensions directory for loading extension assemblies from.<br>• Allow filenames which have '...' as part of the name(three dots).<br>• Many new Javascript Server-Side (JSS) features.<br>• New maximum connections per user setting.<br>• Fixed issue starting CompleteFTP manager on .NET 4.x-only installations on machines with non-English Windows. Manager would terminate with a FileNotFoundException referencing CompleteFTPManager.resources.<br>• Fixed issues with initial blank page appearing after logging in via HTTP/S.<br>• Fixed "Couldn't create unsigned certificate" exception that occurred on install with Windows Server 2003. |
| Version 8.1.6<br>(11 February 2015) | • Use SHA-256 for certificate requests (CSRs) and for generating self-signed certificates.<br>• Can now downgrade the edition automatically when applying purchased activation key.<br>• Added lrename and ldelete to FTP scripting engine, and direct debug output to server log.<br>• Add ability to export IP filter data from manager.<br>• Modify installer so that CompleteFTP service is not removed when installer is run (if it exists).<br>• Tidy overview panel. |
| Version 8.1.5<br>(16 January 2015) | • Permit USER/PASS commands to be sent after logging in as per RFC 959.<br>• Drag and drop for IE in the web file manager.<br>• Fixed short timeout for starting downloads in web file manager.<br>• Fix Gateway problem of closing the FTP connection when accessing a gatewayed FTP server via HTTP.<br>• Fixed bug with default domain being cached when authenticating via automatic Windows users.<br>• Improve memory management for SCP transfers of large files.<br>• Fix OnChannelClosed() exceptions in log in SFTP.<br>• Fix unusual stability issue reported by a user.<br>• Fixed some URLs in the manager application. |
| Version 8.1.4<br>(5 December 2014) | • Improve performance of SFTP under heavy load, and generally reduce memory consumption.<br>• Include client's IP address in logging.<br>• Improve TLS session resumption.<br>• Fix disabling of SSL renegotiation.<br>• Store logging config in separate file so that it doesn't get overwritten on reinstall.<br>• Resizable panels in the group view, process trigger and email notification panels.<br>• Undeprecated IUserInfo.HomeDirectory.<br>• Fixed pscp bug where "Lost connection" was seen.<br>• Fix HTTP/S intermittant download failures.<br>• Fix gateway OutOfMemory error for large transfers. |
| Version 8.1.3<br>(22 October 2014) | • SSL 3.0 is now disabled by default for FTPS and HTTPS. This was motivated by the POODLE vulnerability in SSL 3.0.<br>• Added hmac-sha2-256 and hmac-sha2-512 support to SSH/SFTP.<br>• Optimize use of SMTP client for the server so that email triggers scale better.<br>• Fix minor installer issues - some files weren't being deleted on uninstall, and the manager is now detected if running when the installer is started.<br>• Ensure that the Admin site is always listening on localhost no matter what configuration changes are made.<br>• Fixed an SFTP bug that caused problems for file transfers in the multi-protocol gateway. |
| Version 8.1.2<br>(26 September 2014) | • Add the ability to prevent files with name matching set filemasks from being uploaded or renamed to.<br>• No longer a restriction on number of gateways that can be set up.<br>• Improved error-reporting in web file-manager.<br>• Updated the edtftp.exe scripting binary - adds some new features.<br>• Fixed problems with multi-file upload in web file-manager ('invalid backend response' error).<br>• Fix hanging session problem for users with disk quota.<br>• Fixed "The path is not of a legal form" error in process trigger.<br>• Fixed bug in adding multiple non-Windows users, which added a generated username if the user was already present. |

| | |
|---|---|
| Version 8.1.1<br>(29 July 2014) | • Change from superceded OID for sha1withRSA to PKCS#1 OID in certificate request generator.<br>• Fix bug where HTTP SSL ciphers couldn't be disabled.<br>• Fix problem of Filezilla 3.6.x and later versions getting a parse error on newly generated self-signed certificates in FTPS.<br>• Now rejects client-initiated SSL renegotiation.<br>• Fixed cross-site scripting security hole.<br>• Fix quota bug for random write access.<br>• Fix problem where clients that stop transmitting during transfers but keep connection open do not time out.<br>• Flx so that Chrome FTP works correctly (sends two PASV commands).<br>• Rationalise macros across email & process triggers.<br>• Fixed search in CompleteFTP manager.<br>• Fixed bug where HTTP failed on a site if anonymous user was disabled for that specific site.<br>• Fix Elfinder download bug which occurs when home is set to root. |
| Version 8.1.0<br>(24 February 2014) | • Numerous improvements to the CompleteBox file-sharing client, including file synchronization.<br>• Multi-protocol wrapper now supports publickey authentication for SFTP.<br>• SSH user authentication methods can now be specified on a per user basis in the Enterprise Edition.<br>• Discard changes button in the manager client.<br>• Search added to User Guide, and PDF version included. |
| Version 8.0.1<br>(12 December 2013) | • Add HTTP PUT.<br>• Fix bug where some users received an "Index does not exist" error message in the CompleteFTP manager when deleting a folder.<br>• Fix CompleteBox right-button click from Windows Explorer stopped working bug. |
| Version 8.0.0<br>(2 December 2013) | • Added support for file-sharing, via a file-sharing client, CompleteBox.<br>• Added the ability to write web-apps in Javascript, making CompleteFTP a powerful platform for developing file-oriented web applications.<br>• Improved ability to download diagnostic information for support.<br>• Significantly improved HTTP support.<br>• Many minor enhancements and bug fixes. |
| Version 7.4.0<br>(4 October 2013) | • SSH local port forwarding added, so that SSH tunnels can be set up.<br>• Add new SSH admin shell commands: folderadd, foldermodify, folderdel.<br>• Added timestamp preservation to scp.<br>• Added SSL session resumption to FTPS.<br>• Implemented rename in the Gateway.<br>• Fixed a connection leak in the Gateway.<br>• Fixed incorrect key fingerprint displayed for user keys.<br>• Implemented FXP_FSETSTAT in SFTP.<br>• Fixed scp file overwrite bug.<br>• Fixed bug where RSA host key length was only 1024 in new sites - now 2048 bits.<br>• Manager does not need to be running as administrator to import a new certificate and PVK file.<br>• Fix disk quota bug.<br>• Fix cyclic error message when using cp and mv commands in the SSH shell to copy a file into a subdirectory.<br>• Ensure allowed protocol list is viewable in Standard Edition. |
| Version 7.3.0<br>(3 June 2013) | • Addition of an "Allow always" category in IP filtering. These IPs can't get autobanned.<br>• Allow numeric ranges in IP filters.<br>• In SFTP, if file exists don't send an error in response to FXP_STAT if size or last modified time can't be obtained. This helps some Unix SFTP clients to work correctly.<br>• Improved connection establishment performance by using cached local IPs.<br>• Fixed bug where HTTP session timeout of 0 was not infinite.<br>• Fix problem in email triggers where duplicate emails were sent under certain circumstances.<br>• Move jquery code to local disk to prevent security error.<br>• Call Dispose() on extensions. |
| Version 7.2.1<br>(23 April 2013) | • Now Windows Server 2012 certified.<br>• Improved installer to better detect .NET versions.<br>• Fixed bug in SSH authentication when not all auth methods enabled. |

| | |
|---|---|
| Version 7.2.0<br>(21 March 2013) | • Added HTTP file management - a Javascript file manager supporting upload, download, file viewing and more. Professional and Enterprise Editions only.<br>• Added support for Windows RODCs when listing users (previously needed to be writable).<br>• Enabled public key authentication to be used with Automatic Windows Users (the password must still be supplied).<br>• Database authentication now supports use of password salts.<br>• Allow admin user-name change in Standard and Professional Editions.<br>• Make sure # can be used in file-names in HTTP.<br>• Minor display bug fixes in the CompleteFTP manager.<br>• Documented how to achieve syslog integration. |
| Version 7.1.2<br>(5 February 2013) | • Unlimited-size, unbuffered HTTP uploads. Previously uploads were capped at 20MB because they were buffered in memory before being written to disk.<br>• Added support for customized and automated installation.<br>• Corporate Editions now come with an optional universal activation key (so that online activation is not required). |
| Version 7.1.1<br>(18 December 2012) | • Performance enhancements.<br>• Fixed SFTP bug where if first_kex_packet_follows=true is set, the guess is handled incorrectly.<br>• Fixed "Server offline" bug, where if the server is taken offline in the manager, it can't easily be put online again. |
| Version 7.1.0<br>(26 November 2012) | • Added support for customizable HTTP listings.<br>• Added support for remote ZIP-file navigation. ZIP-files on the server can now be mounted as a directory, changed into, and individual files retrieved (write is not supported as this stage).<br>• Added support for SSH keypair generation for users.<br>• Make restarts of file downloads possible for > 2GB restarts (in FTP and FTPS).<br>• Login/logout events in HTTP.<br>• Add Groups to ISession for use in extensions.<br>• Disallow slashes in virtual folder-names and prevent duplicate folder names. Add hidden folder flag for virtual folders.<br>• Allow CoreFTP to use MDTM in a non-standard way for compatability.<br>• Add option to exclude admin logging in log-view<br>• Make virtual directories non case-sensitive.<br>• Fix broken audit log, which wasn't logging anything. |
| Version 7.0.1<br>(5 October 2012) | • Make CompleteFTPServer.dll strongly named.<br>• Add the FilenameOnly listing format for FTP.<br>• Enable macros to be placed in the names of virtual folders.<br>• Fixed trial installation problem where the error "Key not valid for use in specified state" was seen.<br>• Fixed bug applying Windows permissions of auto-created folders.<br>• Fixed folder re-ordering bug that occurred when renaming folders. |
| Version 7.0.0<br>(20 September 2012) | • Major new feature - cluster support in Enterprise Edition!! This means configuration changes on the primary server in the cluster are automatically propagated to the secondary servers. Very useful for failover and load balancing.<br>• Added ability to add users via a script.<br>• Fixed listings so if a single file was specified, its details are shown.<br>• Added IsValidRSAKey and IsValidDSAKey methods to Authenticator so that users can validate public keys themselves as part of implementing an authentication extension. |
| Version 6.4.2<br>(23 July 2012) | • Groups can now be defined for database users by providing an additional field in the query that supplies a comma-separated list of group-names.<br>• Ensure any temporary script files are created by the process user.<br>• Make standard RSA keylengths 2048 bits (increased from 1024 bits).<br>• Fix bug when trying to log in as a disabled user (IsAnonymous NRE).<br>• Fix security vulnerability in directory navigation.<br>• Fix potential security vulnerability in SSH DSA signing. Thanks to Nadia Heninger for pointing this out.<br>• Fixed bug that made using multiple email addresses as targets for email triggers difficult to use. |
| Version 6.4.1<br>(28 May 2012) | • Modify response when anonymous user is disabled so that Internet Explorer pops up its login prompt.<br>• Fix bug where PVK files generated during CSR creation failed to import when a new certificate was being imported.<br>• Fixed bug in when adding multiple Windows users. If UseHomeForAllSites was not selected then would get "Please make sure every enabled site has a home-folder defined." |

| | |
|---|---|
| Version 6.4.0<br>(7 May 2012) | • Added support editing of properties of multiple users in a single operation. Users can be selected by group membership and wildcard filter. Users may be added to groups when they're being added using the 'add multiple non-Windows users' form.<br>• Group membership is now shown in the user-list in the Users panel.<br>• Users can now be set an expiry date beyond which they cannot log on.<br>• Added support for the generation of Certificate Signing Requests (CSRs) via the CompleteFTP manager.<br>• Made it easier to implement custom authenticators, and added support for doing public key authentication.<br>• Added support so that SSH/SFTP users can change their passwords if this is permitted.<br>• Added support in custom authenticators for forcing password changes for SSH/SFTP users.<br>• When using FTP scripts as process triggers, sometimes they fail with the error "The directory name is invalid". Now fixed. Also upgraded script engine.<br>• Fixed bug where automatic Windows users failed to authenticate in HTTP/HTTPS.<br>• Permit paths to contain ':'.<br>• Fix bug where the manager overview panel fails to restore after minimizing. |
| Version 6.3.0<br>(27 February 2012) | • Add HTTP to the multiprotocol gateway (Enterprise Edition only).<br>• Add new macros, TIMESTAMP:format and TransferStatus for use in events.<br>• Create new email notifications and process triggers by copying existing ones.<br>• Redirect to user's home directory when logging in via /login page in HTTP.<br>• System folders can now be viewed in folder browser in folder tab of manager.<br>• Minor GUI improvements in the manager.<br>• Fix bug where encrypted passwords were enabled and changing the admin password meant they could not be decrypted (and resulted in an error being displayed).<br>• Fix trigger macro bug where WindowsPath and WindowsFolder macros were returning incorrect values when the home-appears-as-root feature was enabled.<br>• Fix bug in email triggers where "Notify on Error" could not be saved.<br>• Fix problem with license.exe - error message was "Could not load file or assembly 'System.Data.SqlServerCe, Version=3.5.1.0".<br>• Fixed file-system permission problem, which applied incorrect folder permissions for sub-folders if "User's home appears as root" was enabled. Previously the permissions of the user's home folder would be applied to all subfolders, overriding any that may have been defined for these. |
| Version 6.2.0<br>(28 December 2011) | • Major Enterprise Edition feature added - the Gateway. This allows CompleteFTP to act as a gateway to other FTP servers - translating protocols in the process.<br>• Added ability to define custom MIME types.<br>• Permit non-Windows users to use the SSH terminal.<br>• Minor bug fixes, and tweaks to manager GUI.<br>• Fix bandwidth limiting bug in SFTP.<br>• Fixed bug where invalid event data could hang subsequent events. |
| Version 6.1.0<br>(20 November 2011) | • Introduced overview panel for Complete FTP Manager.<br>• Added support for HTTP FORM uploads.<br>• Make SFTP the default protocol for the Complete FTP Manager for new installations.<br>• Add flag to enable support of backslashes in paths.<br>• Fix HTTP session bug that can result in multiple authentication prompts.<br>• Simplify installer options and operation.<br>• Fix bug re client certificate authentication. Enforce admin permissions for managing the trusted certificate store. |
| Version 6.0.0<br>(23 August 2011) | • Major Enterprise Edition feature: added support for multiple sites.<br>• Added HTTP directory listings (Professional and Enterprise Editions).<br>• Fixed bug preventing change of directory with a relative path into a virtual dir.<br>• Fixed bug giving inconsistent timestamps for virtual dirs.<br>• Fix bug where administrator password could not be changed if SFTP protocol was chosen.<br>• Added support for multiple deletion of folders and users.<br>• Improved memory management significantly.<br>• Allow indefinite auto-bans (i.e. until next server restart).<br>• Allow additional administrative users.<br>• OPTS UTF ON/OFF added for FTP.<br>• Make contextual help hidable. |

| | |
|---|---|
| Version 5.2.1<br>(18 May 2011) | • Introduction of Enterprise Edition with support for custom extensions for authentication, events, site commands and the file system.<br>• If AUTH argument not understood the client can now respond with USER if FTPS is not mandated.<br>• Now examines the first character of the TYPE argument and compares with I or A rather than comparing the entire argument string. |
| Version 5.2.0<br>(27 April 2011) | • When adding multiple Windows users a single home directory can be now be specified (usually a macro folder).<br>• Added read-only flag for users. If a user is readonly then uploading (and other write operations) is not permitted for that user regardless of what file permissions are set.<br>• Make FEAT more RFC compliant (remove '211-' in front of each feature in the list).<br>• More firewall friendly for FTP active mode transfers (server binds to FTP data port 20).<br>• FTP commands are now uppercased to cater for clients that send 'pasv' rather than 'PASV'.<br>• Improved installer. Status updates are only displayed for certificate and key generation if they are actually being generated.<br>• Fixed bug caused by clashes between naming of virtual file-system folders and Windows file-system folders. Windows folders now take precedence.<br>• Prevent a file being closed prior to last FXP_WRITE by queuing the message (also for FXP_SETSTAT).<br>• Fix bug limiting the number of email notifications (again!), and a bug where changed notifications were saved to the config database but not immediately applied.<br>• Fix bug preventing notifications for HTTP.<br>• Fix SFTP authentication bug re enabled flag.<br>• Fix Windows user directory traversal bug. |
| Version 5.1.1<br>(15 March 2011) | • Fixed bug where FTP connection would be broken by the passive wait timeout if there was a error while getting a listing or transferring in passive mode.<br>• Fix bug in "Admin Access" configuration re error message "The value for column "ExtUserID" in table "Site" is DBNull".<br>• Fix bug in email and process trigger filename filters, where wildcards were being treated as regex's. |
| Version 5.1.0<br>(21 February 2011) | • Permit export of server's private key to file (only if manager is running locally).<br>• Fix for Fetch (Mac OSX), which can't cope with FXP_READ replies that are out of order.<br>• Fix for scp when "User's home folder appears as root" is set.<br>• Fix in FTP so that 550 is returned when attempt made to delete a non-empty directory.<br>• Support for aes128-ctr, aes192-ctr and aes256-ctr in SFTP added.<br>• Support for diffie-hellman-group-exchange-sha1 and diffie-hellman-group-exchange-sha256 in SFTP added.<br>• Fixed bug re anonymous user being permitted when the anonymous user was disabled (but anonymous enabled in Settings).<br>• Fixed bug with multi-line welcome text in FTP when product details are hidden (Windows 7 ftp.exe hung).<br>• Automatic backup of configuration file on a daily basis (if altered).<br>• Fix for infrequent occurrence of server stopping listening. |
| Version 5.0.0<br>(20 December 2010) | • Now supports HTTP and HTTPS.<br>• Added support for external users via an OLEDB connection.<br>• Added support for bulk user imports.<br>• Process triggers now support FTP scripts.<br>• Now supports TLS 1.1 for FTPS.<br>• Handles anonymous user logins in a more standard manner for FTP.<br>• Improved scalability re simultaneous connections.<br>• Add login and logout event auditing.<br>• SITE commands now case insensitive.<br>• Listeners are started/restarted appropriately when "Apply Changes" is selected in the manager and protocols have been enabled or ports changed.<br>• Can now hide product and version number in welcome messages/banners, for PCI compliance.<br>• Fixed error with FTPS implicit where AUTH is (unnecessarily) required to be sent by the client. |

| | |
|---|---|
| Version 4.1.0<br>(10 September 2010) | • Pro version now supports multiple SSH keys per user.<br>• Fix bug where timestamp of newly uploaded file can be incorrectly set.<br>• Implicit FTPS now supported.<br>• SSL email servers now supported for notifications.<br>• SSH exec now supported (for the same commands available in the SSH shell).<br>• Fix bug that allowed guest users access if guest is enabled on the server machine.<br>• Fix bug limiting the number of email notifications.<br>• Improve server memory management. |
| Version 4.0.3<br>(1 June 2010) | • Security fix to disallow paths containing '...' and ':'. This prevents unauthorised directory listings and transfers from outside a user's home directory.<br>• Fix so that Powershell scripts are piped directly into Powershell rather than via a temporary file.<br>• Minor GUI fixes to process trigger setup. Also path filter was not being used in process trigger - now fixed. |
| Version 4.0.2<br>(30 April 2010) | • Domain name filters added. |
| Version 4.0.0<br>(16 April 2010) | • Extensive real-time charts for performance monitoring. (Professional Edition only).<br>• Added Process Triggers - the ability to launch processes on the server when events such as uploading or downloading occur (Professional Edition only).<br>• SSH terminal access added for Windows users (Professional Edition only). Includes ability to execute processes on the server.<br>• Disk quotas introduced (Professional Edition only).<br>• Bandwidth quotas introduced (Professional Edition only).<br>• Can now create a new folder from within the add-user forms.<br>• Fix bug where creating a directory failed in WinSCP (when using SCP only).<br>• Bug fix where it was reported that there was an unhandled exception causing the service to terminate.<br>• Fixed bug where changing directory using forward slashes permitted navigation out of user's directory.<br>• Fixed problem with determining AD group membership.<br>• By default a maximum of 10 anonymous connections are now permitted (prevents someone denying service by connecting as anonymous many times. |
| Version 3.3.0<br>(22 February 2010) | • Add support for autobanning of IP addresses (configurable in Professional Edition). If a client IP has too many authentication failures, it is automatically banned for a period.<br>• Add support for WinSCP with the SCP protocol (already works with WinSCP/SFTP).<br>• Some minor SCP fixes.<br>• Fix manager logging problem - now logs correctly to ApplicationData (for the user).<br>• Fix manager problem where new filters could not be added for the administration server. |
| Version 3.2.2<br>(2 February 2010) | • Fix bug in handling of invalid SSH_FXP_EXTENDED messages.<br>• Add filtering for reducing the number of users returned from large active directory domains.<br>• Can now install the manager only for remote administration from another machine (production install only).<br>• Fixed upload bug in SCP resulting from an infrequent race condition.<br>• Increased minimum SSH window size to 64K to fix problem uploading with cURL.<br>• CompleteFTP manager now logs to the same directory as the server.<br>• Fixed installer bug whereby it always asks if the existing configuration should be kept even when it doesn't exist. |
| Version 3.2.1<br>(18 December 2009) | • Fix "User's home folder appears as root" bug which prevented users from logging in when this flag is set (SFTP only).<br>• Code sign the installation executable. |

| | |
|---|---|
| Version 3.2.0<br>(8 December 2009) | • By default permit all Windows users to logon without explicitly adding them to the user database (see Settings). This can be restricted to users in active directory and/or local groups<br>• Add option for home directory to appear as root ("/").<br>• Add new permissons to Pro version: list folder, delete, create folder permissions. Make Windows and non-Windows permissions distinct.<br>• Added license import/export command-line tool.<br>• Permit non-Windows users to change their passwords (disabled by default).<br>• Fix to prevent error when Cyberduck uploads files.<br>• Fixed key re-exchange bug when kex initiated by clients.<br>• Fixed Blowfish bug in SFTP.<br>• Can now optionally delete the configuration on reinstalling.<br>• When making a license request, allow copy to clipboard for separate emailing.<br>• Allow admin to highlight and kill multiple connections. Auto-refresh of connection view. |
| Version 3.1.2<br>(26 October 2009) | • Added new folder-type called 'Network Folder', which supports UNC paths.<br>• Fixed erroneous license warning from Manager when used remotely.<br>• Fix idle timeout problem causing SFTP sessions to never time out if disconnected abruptly.<br>• Fix 'All users (read)' folder permission in Standard edition.<br>• Allow path to config.sdf to be specified on CompleteFTPConsole.exe command-line.<br>• Better error-handling in Windows file-listings.<br>• Fixed error in new-user form causing half-configured user to be added when the user-root directory does not exist.<br>• Improved login error messages. |
| Version 3.1.1<br>(16 October 2009) | • Fixed problem where not all active directory users were listed if greater than 1000 users.<br>• Fixed problem where failed to login domain users with non-alphanumeric chars in username.<br>• Fixed problem where failed to log in local Windows users that were also on a domain.<br>• Fixed problem where certificate parsing failed if there was a comma or semicolon in the subject.<br>• Fixed problem where multiple Windows users could not simultaneously share MyDocuments.<br>• Stop CFTP Manager minimizing to tray.<br>• Stop Windows users from accessing other home directories. |
| Version 3.1.0<br>(4 September 2009) | • User-specific protocol settings.<br>• Reduce memory usage during log file viewing.<br>• Stop scp client from hanging if SCP is disabled in the server and a command is attempted.<br>• Fixed error where target filename could not be specified in scp.<br>• Improved passive mode port range handling.<br>• Logging improvements.<br>• Fixed bug preventing deletion of e-mail notifications.<br>• A variety of minor GUI fixes. |
| Version 3.0.1<br>(7 August 2009) | • Fix logging bug where can't change log level to Debug.<br>• Fix notification bug where can't delete last notification.<br>• Fix incorrect log file location (Program Files -> Program Data).<br>• Fix SCP bug where copy failed if target was a filename.<br>• Minor performance tuning changes. |
| Version 3.0.0<br>(30 July 2009) | • Added support for SCP (Professional Edition).<br>• Permission editing added (more sophisticated version in Professional Edition).<br>• Email notification for a variety of events such as file upload, download, delete (Professional Edition).<br>• Auditing (i.e. tracking all changes to files - Professional Edition).<br>• FTPS client certificate verification (Professional Edition).<br>• IP filtering added - accept/reject connections based on IP address (Professional Edition).<br>• Can now specify the network interfaces to listen on.<br>• No longer disconnects channel when client sends SSH_CHANNEL_EOF.<br>• Added support for MODE S (which is the default).<br>• Fix re file overwrite problem. |

| | |
|---|---|
| Version 2.2.1<br>(24 June 2009) | • Now closes console gracefully.<br>• Fixed bug whereby a disabled user could still log in via SFTP.<br>• Fixed bug where newly generated SSL certificate failed to save to the config.<br>• Fixed auto-add user checkbox in 'Add Windows user form'.<br>• Fixed bug where when a new activation key is entered, it is saved but the manager appears to show that it has not.<br>• Manager now uses compression to speed up log file transfers.<br>• Manager connection editor now allows clearing of properties.<br>• Added extra SFTP and FTPS properties to Manager connection settings.<br>• Improved handling of socket accept errors. |
| Version 2.2.0<br>(2 June 2009) | • Introduction of professional version supporting Windows domain users.<br>• Now works on x64 machines.<br>• MODE Z compression added for FTP and FTPS (also OPTS MODE Z).<br>• Can now directly download log files to local disk.<br>• Fixed SFTP authentication bug, where if the username was supplied in the incorrect case, public key authentication failed.<br>• Users can no longer list /home, i.e. they can't see what the other user directories are.<br>• Administrator can retrieve user list (Professional version only).<br>• Manager now displays the version number.<br>• Fixed FEAT bug (indenting too many spaces for each feature listed).<br>• Fixed timeout that occurs in passive mode with large transfers if the passive timeout is set.<br>• Fixed bug whereby SFTP admin could only log in once.<br>• Manager now displays any critical server errors when connecting. |
| Version 2.1.2<br>(7 May 2009) | • Can now call TYPE command in between PORT/PASV and RETR/STOR (which some clients do).<br>• Administration functionality can now be performed via SFTP.<br>• Invalid user names aren't immediately rejected, forcing hackers to try authenticating.<br>• Fix for WinSCP problem with files very occasionally being truncated.<br>• Provide alternative for obtaining machine id hash if email client is not installed (for product activation).<br>• Fix problem with using a non-default home directory when creating a Windows user.<br>• Fix resource cleanup bug when client abruptly disappears. |
| Version 2.1.1<br>(23 March 2009) | • Added support for the XCRC FTP command to check file integrity.<br>• Better feedback while installer is running.<br>• Added diffie-hellman-group14-sha1 key exchange method.<br>• Should now work on .NET 2.0 SP1 (previously required a minimum of 2.0 SP2).<br>• Fix problem with generated FTPS certificate being initially invalid. |
| Version 2.0.0<br>(11 February 2009) | • Now offers full support for SFTP (FTP over SSH). Support for password and publickey authentication, RSA and DSA keys etc.<br>• Can now set logging level in the manager.<br>• User's Guide expanded.<br>• Generation of server certificate takes place on the server.<br>• Numerous enhancements to the Manager. |
| Version 1.1.0<br>(12 November 2008) | • Fixed license reading bug - sometimes the license was not read correctly, resulting in the server running in restricted mode.<br>• Fixed user impersonation bug that meant special folders (e.g. My Documents) could not be navigated to.<br>• Database and log files now stored in C:\ProgramData (or equiv) rather than in Program Files.<br>• Default encoding is now UTF-8. Encoding can now be changed.<br>• Added User's Guide.<br>• Improved responsiveness of Windows folder browser.<br>• Set default name of Windows virtual folder to name of actual folder.<br>• Fixed FTPSEnabled property.<br>• Added ability to add multiple windows users in one operation. Fixed NRE when NodeForm was cancelled.<br>• More descriptive message when max number of connections exceeded.<br>• Returns meaningful message when AUTH is used and FTPSEnabled is off.<br>• Rearranged controls and changed text to make it clearer that the settings are the settings of the admin connection and not of the server itself.. |
| Version 1.0.0<br>(10 October 2008) | • First commercial version! After many months of further development and testing.<br>• New manager, better FTPS support, remote administration, too many other new features to explain here. |

| | |
|---|---|
| Version 0.5.1<br>(5 December 2007) | • Fix permitting LIST to accept '-' arguments.<br>• Disable idle time forced logouts by default. |
| Version 0.5.0<br>(13 November 2007) | • First release, supporting FTP and FTPS. |

# Installation

**CompleteFTP** is supplied as an executable Windows installer. It is also possible to customize and automate installation.

To install, double-click on the installer executable, and you will be taken through the steps required to set up CompleteFTP. You may require Administrator permissions to run the installer.

The trial installer installs both the server components and the Complete FTP Manager. The production installer by default installs both, but optionally permits just the Complete FTP Manager to be installed if required, so that it can be used for remote administration of a server installed on another machine.

Note that you will have the option of installing the *Standard Edition*, the *Professional Edition* or the *Enterprise Edition*. More details on how to select the edition can be found here. The trial edition can be upgraded from any lower edition to a higher edition.

Please note that CompleteFTP installs as a Windows Service, running as the local SYSTEM user. It will be started automatically when your computer starts. Once installed, run the Complete FTP Manager to begin adding users.

The trial edition expires after 31 days. Once the trial period is over, the server converts to Developer Mode, and will only accept connections from *localhost* (but will otherwise remain fully functional). CompleteFTP can be used in Developer Mode indefinitely. It may be converted to production mode by purchasing a license.

Once a license has been purchased, a trial or developer mode installation (or a production installation) can be **activated** as described here.

There are detailed getting started tutorials available that explain how to connect with Filezilla and WinSCP.

## Uninstalling

To uninstall CompleteFTP, use one of the methods listed below:

1. Select the ï¿½Uninstall Complete FTPï¿½ menu item from the Tools submenu in the CompleteFTP program group
2. Navigate to the Windows Control Panel, and select "Programs and Features". CompleteFTP will appear in the list of programs. Right click and choose "Uninstall".

## How to run the server

*CompleteFTP* is installed as a Windows service, running as the local SYSTEM user, and by default is configured to run automatically on machine startup. The service can be stopped and started via Control Panel -> Administrative Tools -> Services. Select the CompleteFTP Service. Under more recent versions of Windows you will need to have administrator privileges to do this.

The service can also be started from the CompleteFTP Manager - but only if 1) it has been started on the server machine and 2) it was launched with administrator privileges.

If the CompleteFTP service fails to start, or starts with errors, the most likely problem (other than the lack of administrator privileges) is:

- the standard FTP port, 21, is already in use on the machine
- the standard SSH port, 22, is already in use on the machine
- the standard HTTP (80) or HTTPS (443) ports are already in use on the machine

Using 'netstat -a' in a DOS prompt will reveal what ports are in use. IIS is usually the most likely application already using ports.

If IIS will continue to be used for HTTP and HTTPS, then HTTP and HTTPS will need to be disabled in CompleteFTP - see here.

CompleteFTP can also be run as a console application (rather than as a service). This can often be useful in debugging problems. Select *Run server in console* from the CompleteFTP program menu (with the service stopped) to do so. It is usually necessary to "Run as administrator".

## How to use the manager

The *CompleteFTP manager* controls all settings for the CompleteFTP server (which typically runs as a Windows service).

The manager is started by selecting the CompleteFTP Manager icon from the CompleteFTP menu. Normally, the service must already be running for the manager to connect to it. If the server is not running, the manager will prompt to ask if the CompleteFTP Service should be started. If the manager is running with sufficient permissions, the service will be started. If not, the Windows Control Panel -> Adminstrative Tools -> Services must be used to start the service.

The connect dialog box is brought up when the manager is selected. This allows details of the host that the CompleteFTP service is running on, and the administrator credentials to be entered.

The defaults should generally be used (they are set in installation), and are shown below:

| **Setting name** | **Default value** |
|---|---|
| Server name/IP | localhost |
| Server admin port | 14983 |
| Administrator username | admin |
| Administrator password | [whatever was entered on install] |

Note that more detailed settings for the connection can be accessed via the *Other settings...* link. Normally these should not be modified. The "Save password" checkbox can be checked to save the administrator password so it is automatically entered, and "Show connection log" can be checked if there are problems connecting to help resolve the issue.

Once the CompleteFTP manager is running, all settings can be accessed via the list of panels on the left hand side. These include settings for the Site(s), File System, Users, Groups, Events, Extensions (for the Enterprise edition), Monitoring, Logging and Licensing.

# How to migrate to another machine

At times it may be necessary to migrate a *CompleteFTP* installation to another machine. To do so while preserving settings such as user logins, passwords and home folders, follow the steps below:

1. Copy the files and subdirectories of the data-directory (see below) from the old machine to the same location on the new machine.
2. Install CompleteFTP on the new machine. A later version may be installed on the new machine as the installer will upgrade the configuration.
3. Request a new activation key in the same way that you did when you first activated CompleteFTP (the activation key is tied to the Windows installation, so a new key is required).

On Windows Vista and Server 2008 the data-directory is:

*C:\ProgramData\Enterprise Distributed Technologies\Complete FTP*

On Windows XP and Server 2003 the data-directory is:

*C:\Documents and Settings\All Users\Application Data\Enterprise Distributed Technologies\Complete FTP*

See also How to automatically import a license for details on automating the process of adding an activation key to the configuration (useful in failover scenarios).

Note that if you have hard-coded IP addresses in your old configuration, you will need to modify them appropriately via the CompleteFTP manager once installation on the new machine is complete.

# How to automate installation

The CompleteFTP installer is a standard NSIS-based Windows installer, which uses a GUI to prompt for various options during installation.

Some organizations prefer to automate installations, and use command-line scripts to roll out software. A GUI-based installer is usually not suitable for this - instead a command-line installer that requires no user input is required. This article describes how to create and use command-line installers for CompleteFTP.

There's a distinction between a first-time installer and an update installer. A first-time installer is used only when a completely new installation is required, or when an existing installation is to be completely overwritten. An update installer should be used when an existing installation is to be updated to a more recent version.

The general pattern for creating and using automated installers is to:

1. Install CompleteFTP using the standard GUI installer on one machine (the 'development' machine)
2. Create archives of the required files
3. Un-archive these on another machine (the 'production' machine)
4. Execute various commands to make the software operable on that machine.
5. If necessary, activate the installation with an activation key (AK).

The sections below don't contain the full scripts that must be executed, but rather describe the actions that must be performed.

**First-Time Installation**

A first-time installation is an installation onto a machine that either has no current CompleteFTP installation, or has an existing CompleteFTP installation that is to be entirely overwritten.

An activation key (AK) will be required to activate the installation. Most users developing automated installation scripts will have purchased the Corporate license, which entitles users to a universal activation key. Without using a universal activation key, each installation will later need to be activated manually.

### Development machine: Creating archives

1. Install CompleteFTP using the standard GUI installer on the development machine
2. Configure CompleteFTP as desired using CompleteFTP Manager
3. Activate the installation using the universal activation key
4. Create a ZIP file called `binaries.zip` that contains the directory `C:\Program Files (x86)\CompleteFTP` (including the directory itself)
5. Remove `install.log` and `server\boostrapper.log` from the ZIP-file
6. Create a ZIP file called `data.zip` that contains `C:\ProgramData\Enterprise Distributed Technologies` (including the directory itself)
7. Remove the contents of the `Temp`, `Logs` and `Backups` directories from the ZIP-file
8. Remove any unwanted directories and files from the Users directory

### Production machine: Installation

1. Unzip `binaries.zip` in `C:\Program Files (x86)` (unzip app needs admin privileges)
2. Unzip `data.zip` in `C:\ProgramData` (unzip app needs admin privileges)
3. Locate the `InstallUtil.exe` program - usually in `C:\Windows\Microsoft.NET\Framework\v2.0.50727` or later version
4. Install the CompleteFTP service with the command:
   ```
   InstallUtil.exe "C:\Program Files (x86)\Complete FTP\Server\CompleteFTPService.exe"
   ```
5. Change the startup type of the service to Automatic with the command:
   ```
   reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\CompleteFTP /v Start /t REG_DWORD /d 2 /f
   ```
6. Stop parent directory of user accounts from inheriting permissions from its parent:
   ```
   icacls "C:\ProgramData\Enterprise Distributed Technologies\Complete FTP\Users" /inheritance:d
   ```
7. Remove read permissions for the Users group for the same folder (so that Windows users can't get a list of other users):
   ```
   icacls "C:\ProgramData\Enterprise Distributed Technologies\Complete FTP\Users" /remove:g Users
   ```
8. Each Windows user-account which is registered in CompleteFTP must have permission to their own home-directory, so if there are any such users then use `icacls` to grant them
9. Start service using the command
   ```
   net start CompleteFTP
   ```

**Updating an Existing Installation**

When a production installation is being updated, the new binaries may be copied over the old ones, but the configuration data on that machine must not be overwritten. The main difference between the first-time installation and the update is therefore related to the treatment of the data.

### Development machine: Creating archives

1. Update existing CompleteFTP using the standard GUI installer on the development machine
2. Create a ZIP file called `binaries.zip` that contains the directory `C:\Program Files (x86)\CompleteFTP` (including the directory itself)
3. Remove `install.log` and `server\boostrapper.log` from the ZIP-file

### Production machine: Installation

1. Stop CompleteFTP Service using the command

```
net stop CompleteFTP
```

2. Delete the directory `C:\Program Files (x86)\CompleteFTP` and all its subdirectories
3. Unzip `binaries.zip` in `C:\Program Files (x86)` (unzip app needs admin privileges)
4. Update the configuration database to latest version using the command
   ```
   C:\Program Files (x86)\CompleteFTP\cftpconfig
   ```
5. Perform any configuration modifications (see cftpconfig)
6. Start service using the command
   ```
   net start CompleteFTP
   ```

# How to activate an installation of CompleteFTP

When a production license has been purchased, the trial installation must be *activated*.

This usually involves accessing the *CompleteFTP Activation System* on the Internet, but customers who have purchased Corporate licenses can request a universal activation key which does not require Internet access.

**Activating on a machine with Internet access**

A. Run the CompleteFTP manager and connect to the CompleteFTP server instance that you wish to activate (the server instance - the CompleteFTP Windows service - may be running on another machine). You should run the manager on a machine with internet access (although there is a work around) if you do not have a universal activation key.
B. Select the "Licensing" menu item from the list on the left in the manager, and then select the "Activate your license" link.
C. Enter your purchase reference in the edit box indicated (Step 1), and then open the supplied URL. You will need the login details provided in your purchase email. If the credentials and purchase reference are correct, you will see a web page called "CompleteFTP Activation System".
D. Your activation key should be displayed in the text box, and it can be copied and pasted into the CompleteFTP manager in Step 4. Select "OK" and it should now be activated, and the license details should be displayed.

**Activating on a machine without Internet access**

A. Run the CompleteFTP manager and connect to the CompleteFTP server instance that you wish to activate (the server instance - the CompleteFTP Windows service - may be running on another machine).
B. Select the "Licensing" menu item from the list on the left in the manager, and then select the "Activate your license" link.
C. Enter your purchase reference in the edit box indicated (Step 1).
D. Click the "copy" link in Step 2 and copy this link to a machine that has Internet access.
E. On the machine that has Internet access open the link in a browser. You will need the login details provided in your purchase email. If the credentials and purchase reference are correct, you will see a web page called "CompleteFTP Activation System".
F. Your activation key should be displayed in the text box. Copy this activation key back to the machine on which CompleteFTP is installed.
G. Paste the activation key into the CompleteFTP manager in Step 4. Select "OK" and it should now be activated, and the license details should be displayed.

**Activating with a Universal Activation Key**

A. Run the CompleteFTP manager and connect to the CompleteFTP server instance that you wish to activate (the server instance - the CompleteFTP Windows service - may be running on another machine).
B. Select the "Licensing" menu item from the list on the left in the manager, and then select the "Activate your license" link.
C. Paste the universal AK you have been sent into the CompleteFTP manager in Step 4. Select "OK" and it should now be activated, and the license details should be displayed.

If you need to move an instance of CompleteFTP from one server to another, or are having difficulties with activating, please email us.

## How to upgrade

To upgrade an existing installation, the new production installer is required. This can be downloaded from the EnterpriseDT website **using the link and login details you received when you purchased your license**. You **cannot** use the trial installer that is available on the website to upgrade a production installation. Please contact support if you cannot locate your login details.

To perform the upgrade, exit the CompleteFTP manager application if it is running, and install the new version. The default option is to upgrade your existing configuration to the new version (preserving your settings). You do not need to uninstall prior to running the new installer.

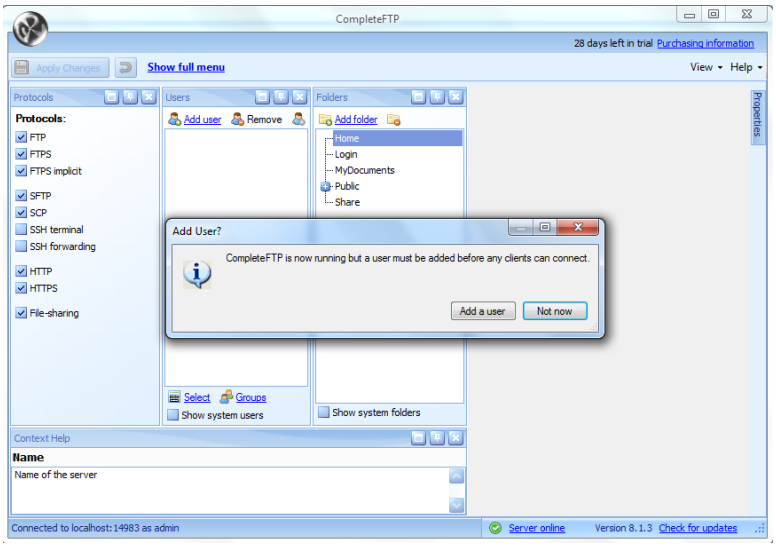
# Getting started with CompleteFTP

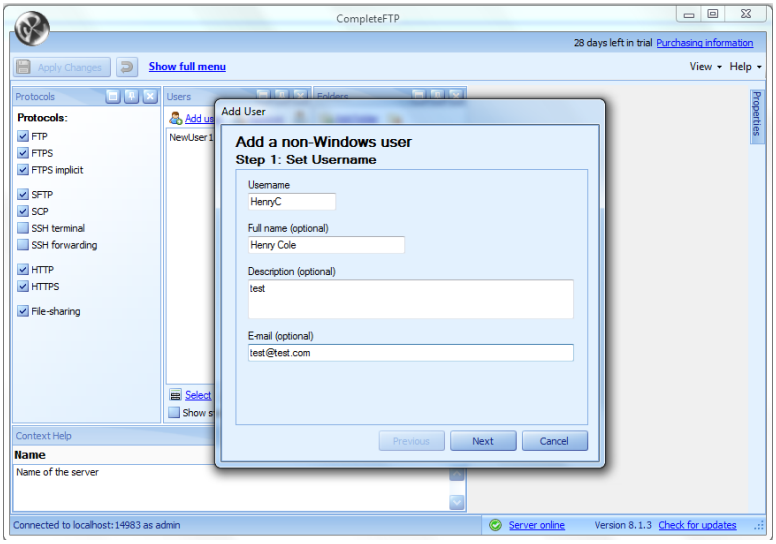
This tutorial explains how to get started with CompleteFTP, our Windows secure file server supporting a variety of secure file transfer protocols, including FTPS, SFTP, SCP and HTTPS.

## Steps

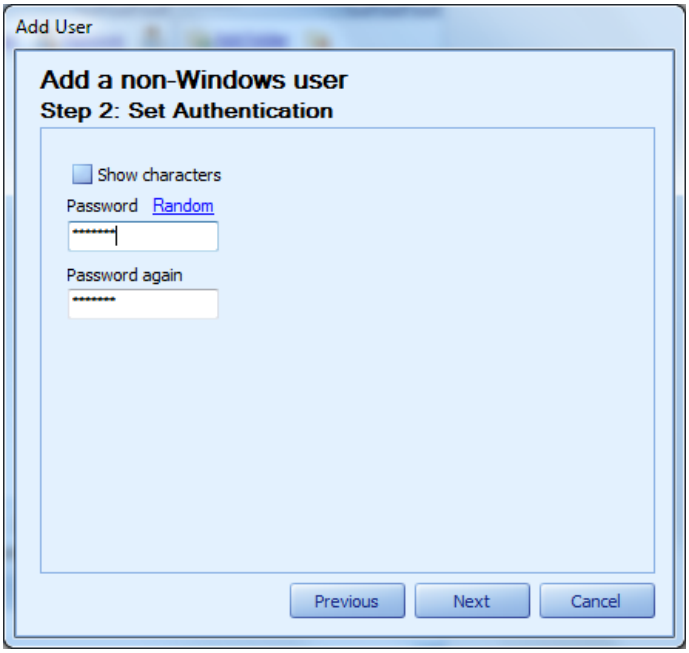
1. The latest Windows trial installer can be downloaded from here. To install, double-click on the installer executable. More details can be found here.

2. After the installation is complete, run the Complete FTP Manager to beginï¿½adding users.

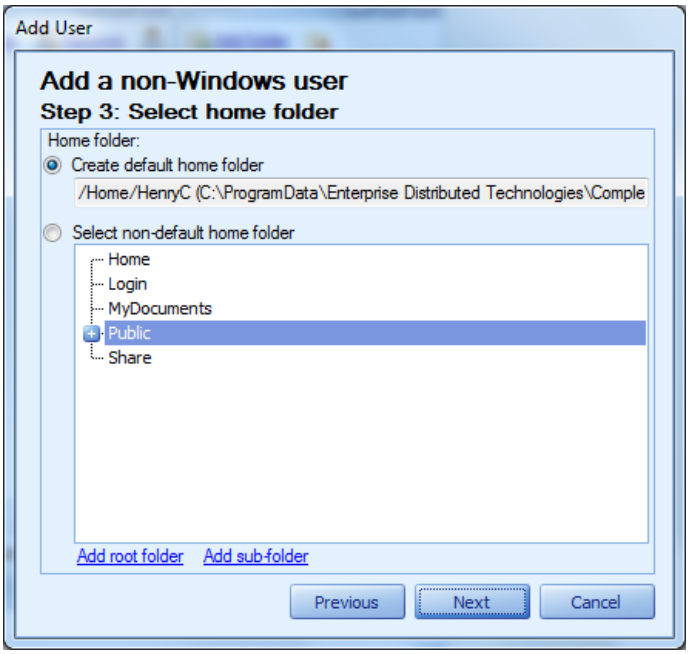


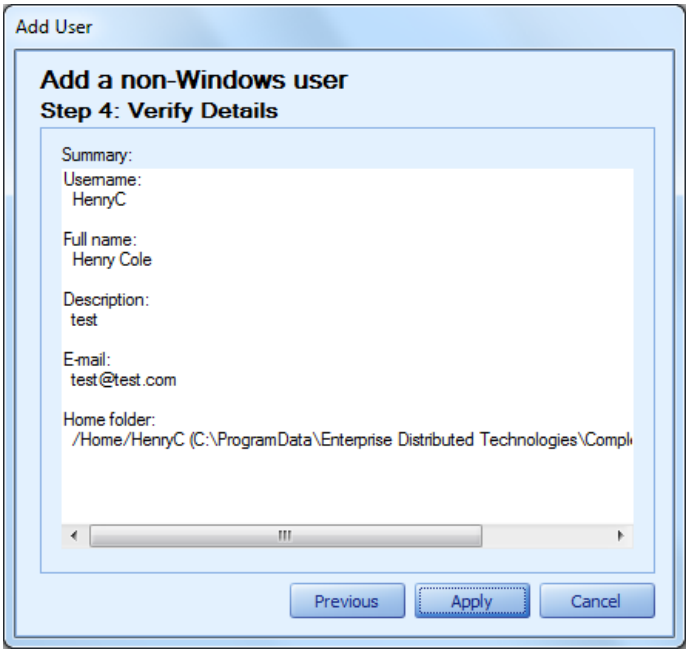

3.ï¿½A password is required for any user account you create. It is advisable forï¿½this to be as secure as possible. Alternatively, use the 'Random' button to quickly generate a secure password.

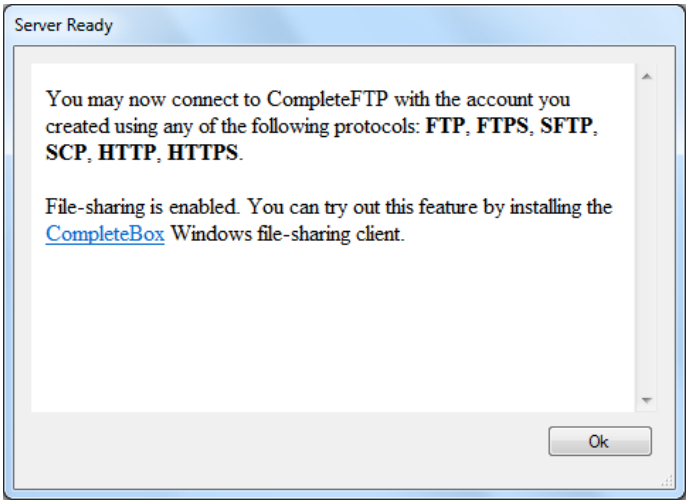



4. Select the home directory where the dataï¿½for this user will be stored. The default is usually the best option,




5. Verifyï¿½the information entered previously and select *Apply*:




Server Ready
You may now connect to CompleteFTP with the account you created using any of the following protocols: FTP, FTPS, SFTP, SCP, HTTP, HTTPS.
File-sharing is enabled. You can try out this feature by installing the CompleteBox Windows file-sharing client.
Ok

6. The server is now available for connection using an FTP, FTPS or SFTP client.

### How to connect

For detailed information on how to connect to this server using popularï¿½clients, seeï¿½our tutorials:

- Connecting with WinSCP
- Connecting With FileZilla

# Connecting to CompleteFTP with Filezilla

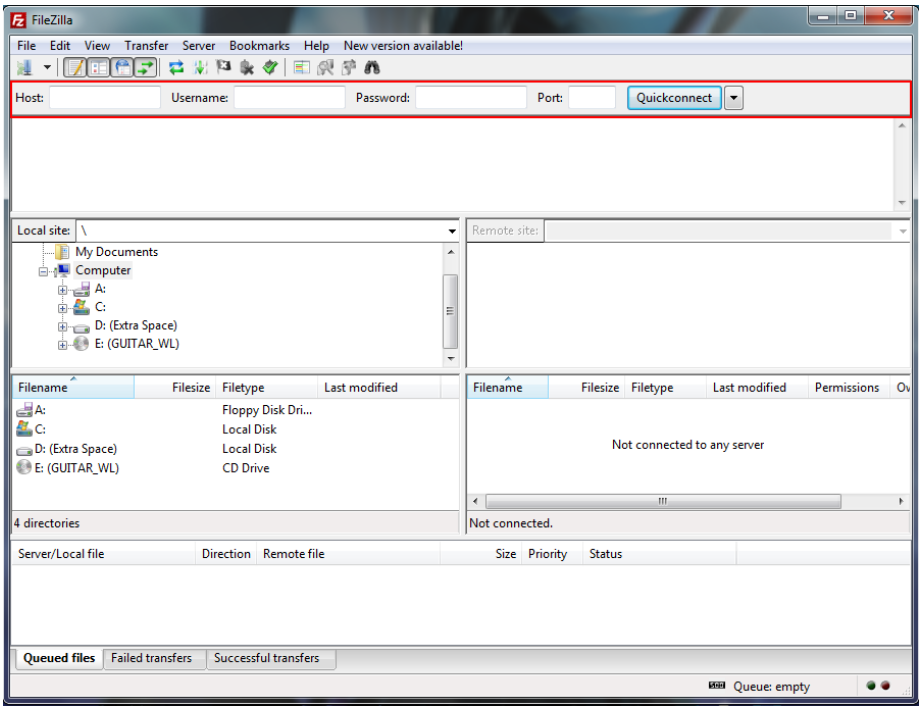
This tutorial demonstrates how to connect to your CompleteFTP server using the popular FTP/SFTP client,ï¿½**FileZilla**, whichï¿½can be downloadedï¿½here.ï¿½**FileZillaï¿½**supports FTP, FTPS and SFTP.

Before looking at this tutorial, make sure you have CompleteFTP installed, and a user created (*seeï¿½here*).

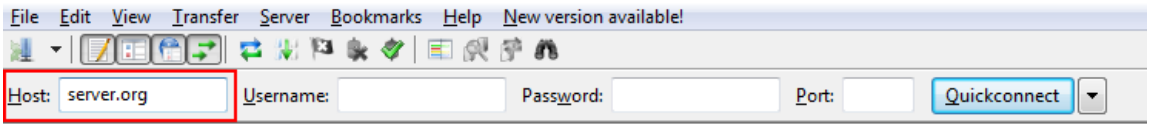
## Steps

1. After installing FileZilla, open up the program.

Nearï¿½the top of the FileZilla window, there is a *QuickConnect* Bar:



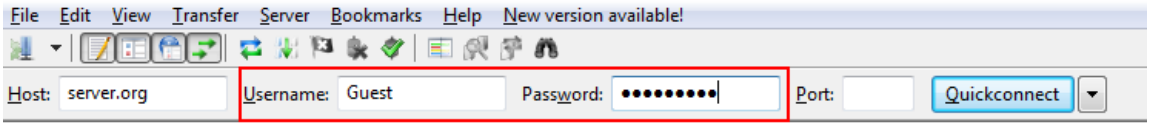

2. Enter your server's addressï¿½into theï¿½*Host* Sectionï¿½(*use 127.0.0.1, or localhost if you are connecting from the same computer you are hosting the server on).*




If you want to connect using FTPS (FTP over SSL), enter the address as 'ftps://server.org'.

If you want to connect usingï¿½SFTP (FTP over SSH) enter the address as 'sftp://server.org' .

3. Enter your users's details into the *Username* and *Passwordï¿½*sections.ï¿½Theseï¿½should be the same details you entered for this user when they were added to CompleteFTP usingï¿½theï¿½**CompleteFTP** Manager (see Getting Started With CompleteFTP)

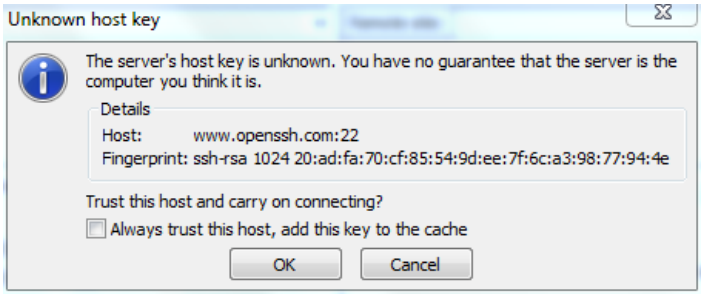


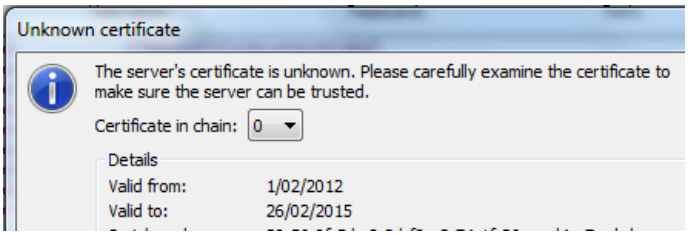

(*you can leaveï¿½the* Port *section blank, unless youï¿½require a specific port number*)

5. Press QuickConnect to connect to the server.

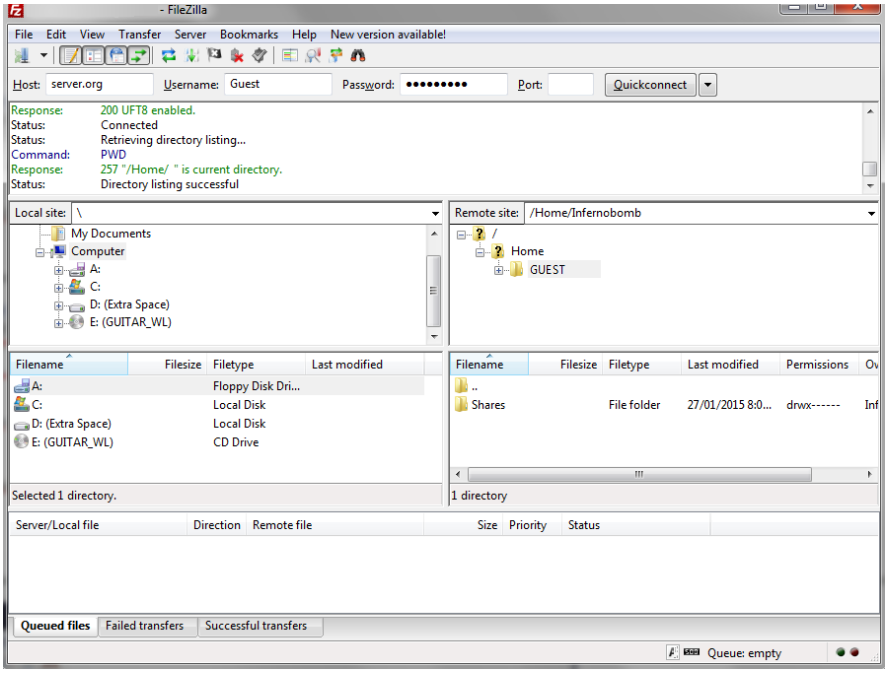
6. If you have chosen SFTPï¿½or FTPS, a warning about the host will appear. This warning is because you have never connected to this server before, and WinSCP does not have its host key (SFTP/SCP) or its certificate (encrypted FTP)- which identifies the server to clients - installed.

Click Yes to the warningï¿½to add the certificate or host key to your cache. Both warnings are displayed below - you will receive only one of them, depending on the protocol used.

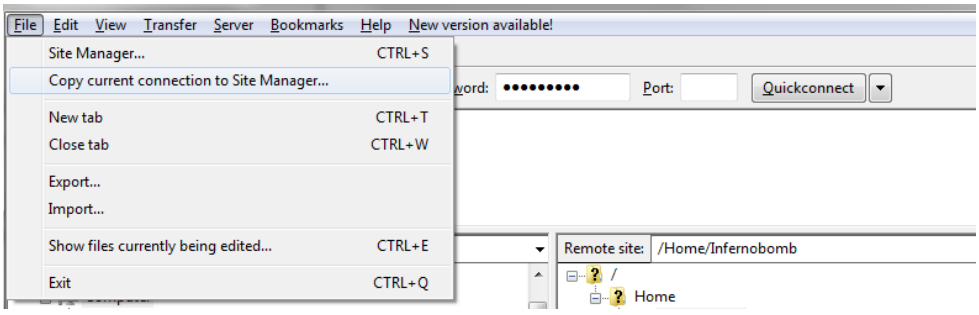

Unknown host key
The server's host key is unknown. You have no guarantee that the server is the computer you think it is.
Details
Host: www.openssh.com:22
Fingerprint: ssh-rsa 1024 20:ad:fa:70:cf:85:54:9d:ee:7f:6c:a3:98:77:94:4e
Trust this host and carry on connecting?
Always trust this host, add this key to the cache
OK
Cancel


Unknown certificate
The server's certificate is unknown. Please carefully examine the certificate to make sure the server can be trusted.
Certificate in chain: 0
Details
Valid from: 1/02/2012
Valid to: 26/02/2015

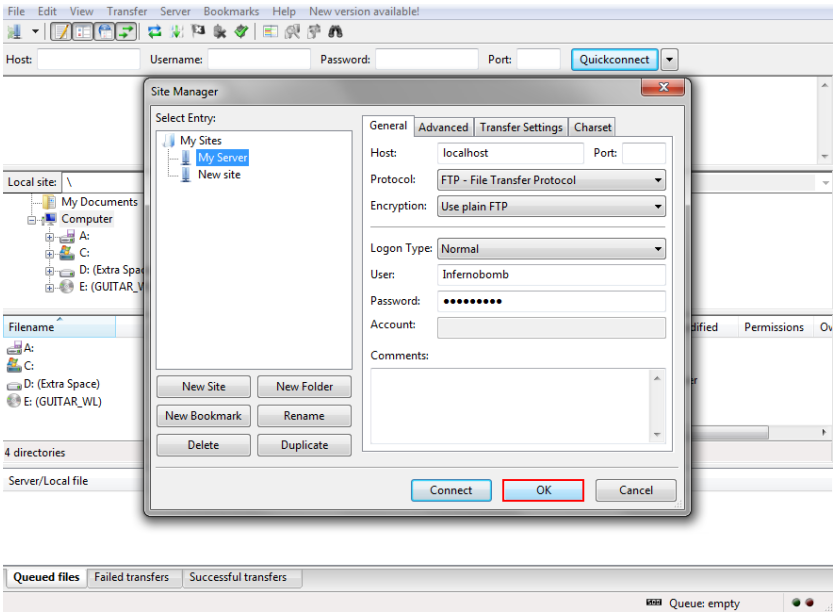
You shouldï¿½now be connected to CompleteFTP through FileZilla, using whatever protocol specified in your URL.




To add the server information to the site manager for later reference, select C*opy current connection to Site Manager*ï¿½inï¿½the File menu;

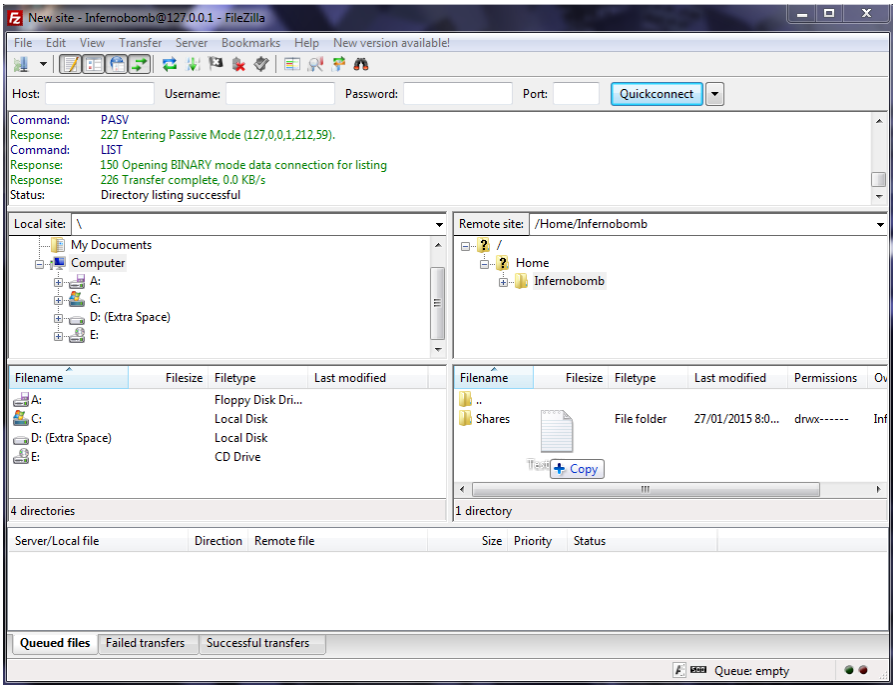



You will find all the important details automatically filled out, leaving you to press OK;




ï¿½Instead of the Quick Connect, you can also use the site manager to select a site and then connect.

To add files to the server, drag and drop from the bottom left panel (your computer's directory) to the bottom right panel (the server's directory). Alternatively, drag and drop directly from your Computer toï¿½the right panel;



# Connecting to CompleteFTP with WinSCP

This tutorial demonstrates how to connect to yourï¿½CompleteFTPï¿½server using the popular FTP/SFTP client,ï¿½**WinSCP**, whichï¿½can be downloadedï¿½here.ï¿½**WinSCPï¿½**supports FTP, FTPS, SFTP and SCP.

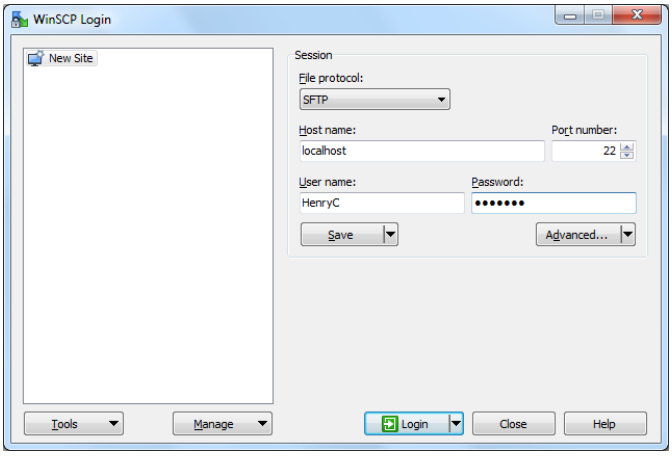
Before looking at this tutorial, make sure you have CompleteFTP installed, and a user created (*seeï¿½here*).

## Steps

1. After installingï¿½and launching WinSCP, the Login window is shown, below.

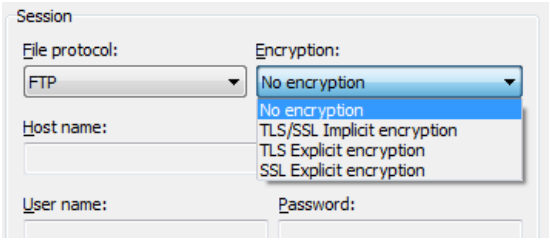
Enter the user name and password of a user that has been createdï¿½in CompleteFTPï¿½previously.

If you are connecting from the same machine that CompleteFTP was installed on, you can useï¿½*127.0.0.1*, or *localhost* for the host name. Otherwise use the hostname of the server. If you are unsure what that is, ask your network administrator.




Now select the protocol to connect with - FTP, SCP or SFTP.

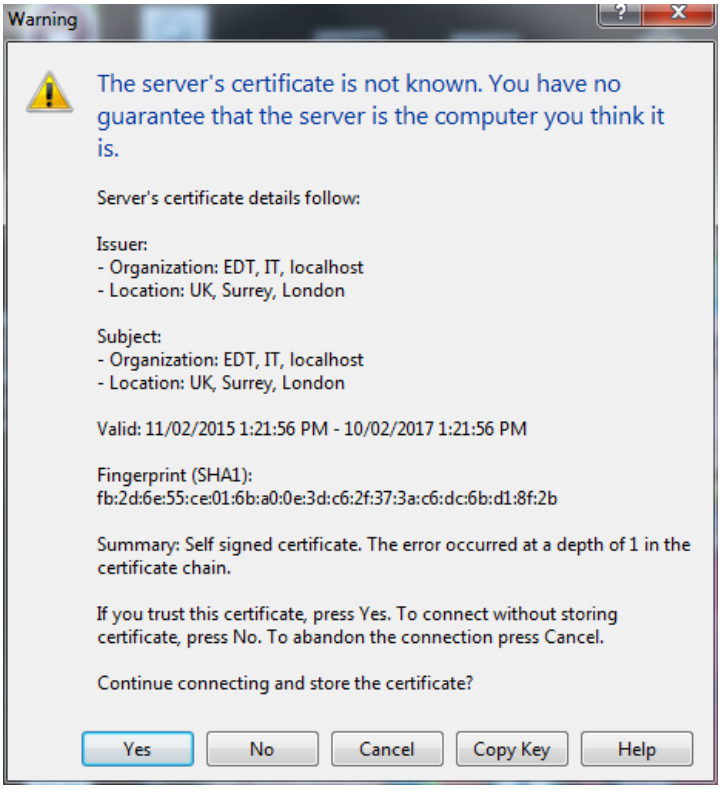
If FTP is chosen, use the Encryption drop-down menu to select what type of FTP you want. If you want plain, unencrypted FTP, select "No encryption". If you want FTPS, which is FTP over a secure connection, select "TLS Explicit encryption".

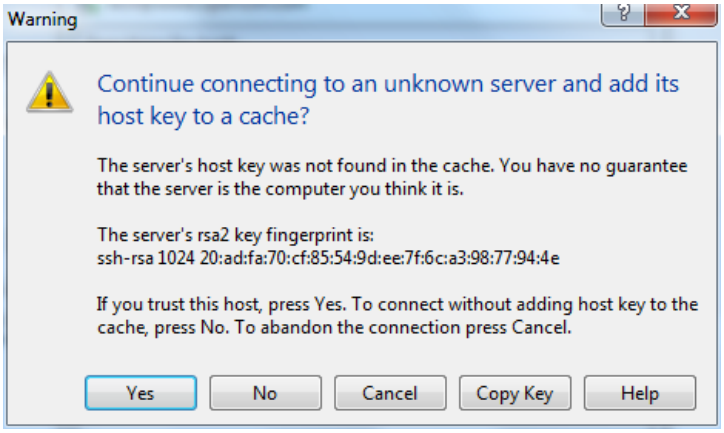



ï¿½Then select "Login" to log into the server.

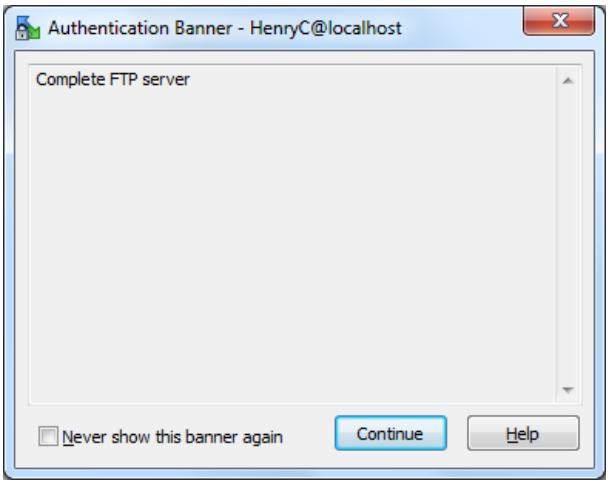
2. If you have chosen SFTP, SCP or an encrypted FTP, a warning about the host will appear. This warning is because you have never connected to this server before, and WinSCP does not have its host key (SFTP/SCP) or its certificate (encrypted FTP)- which identifies the server to clients - installed.

Click Yes to the warningï¿½to add the certificate or host key to your cache. Both warnings are displayed below - you will receive only one of them, depending on the protocol used.

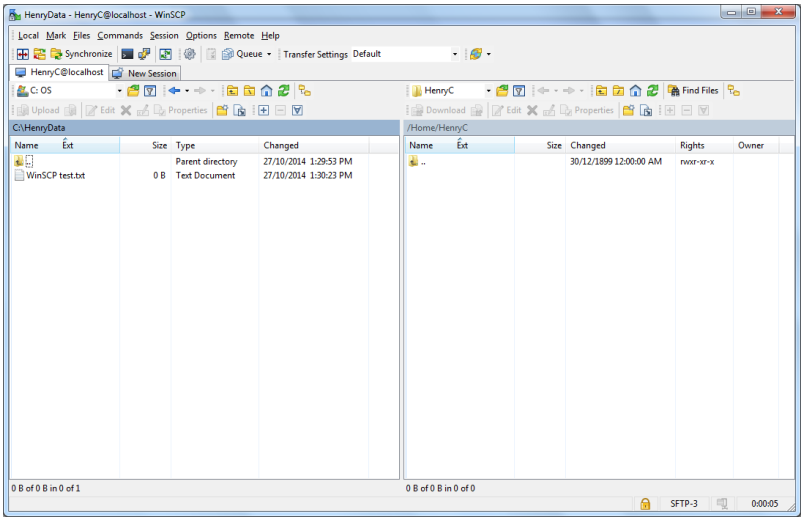

Warning
The server's certificate is not known. You have no guarantee that the server is the computer you think it is.
Server's certificate details follow:
Issuer:
- Organization: EDT, IT, localhost
- Location: UK, Surrey, London
Subject:
- Organization: EDT, IT, localhost
- Location: UK, Surrey, London
Valid: 11/02/2015 1:21:56 PM - 10/02/2017 1:21:56 PM
Fingerprint (SHA1):
fb:2d:6e:55:ce:01:6b:a0:0e:3d:c6:2f:37:3a:c6:dc:6b:d1:8f:2b
Summary: Self signed certificate. The error occurred at a depth of 1 in the certificate chain.
If you trust this certificate, press Yes. To connect without storing certificate, press No. To abandon the connection press Cancel.
Continue connecting and store the certificate?
Yes
No
Cancel
Copy Key
Help


Warning
Continue connecting to an unknown server and add its host key to a cache?
The server's host key was not found in the cache. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 1024 20:ad:fa:70:cf:85:54:9d:ee:7f:6c:a3:98:77:94:4e
If you trust this host, press Yes. To connect without adding host key to the cache, press No. To abandon the connection press Cancel.
Yes
No
Cancel
Copy Key
Help

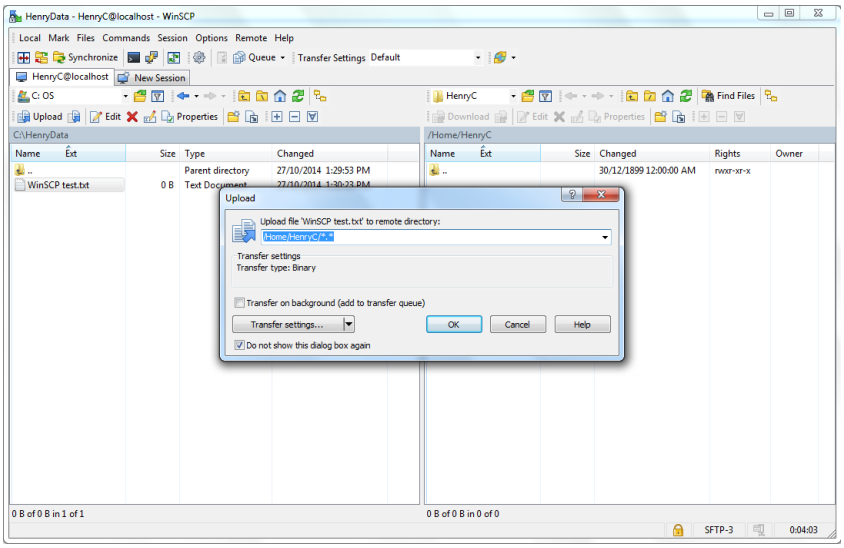
3.ï¿½Next, the server's banner message will be displayed, as below. Select "Continue" to finish connecting. Check "Never show this banner again" to prevent it being displayed again.




4. You are now connected through WinSCP to the CompleteFTP server, and can start transferring.




ï¿½To transferï¿½files to the server, simply drag and drop from the left panel to the right.ï¿½The left panel is your home computer and the right panel is your CompleteFTP server.



## Overview

The Overview panel allows the administrator to combine a series of views within the one panel. Available views include: site, users, folders, properties, performance and connections.

Views may be shown/hidden via the View menu in the top-right of the window. Views can be resized, moved, docked to any edge, combined into tabs and moved into separate "floating" windows. A view may be resized by dragging its edge and moved by dragging its title-bar. While a view is being dragged several icons are shown overlaid on the window; dropping the view onto one of these will dock it to an edge or place it in a tab. The video below illustrates some ways in which panels may be configured:

The properties view shows the properties of the currently selected site, user or folder. If the property view is already visible then it will automatically change as different items are selected. If it is not visible then it may be displayed by right-click on a user or a folder and selected *Properties*.

# Site/Sites

A CompleteFTP *site* consists of a related group of protocols and port numbers - for example the default ports for FTP on port 21 and SFTP on port 22 make up the "Default site" (below). Sites have their own settings, including public keys and certificates. Every installation of CompleteFTP supports at least two *sites* - the 'Admin site' and the 'Default site'.

**Default site**

As described above, the "Default site" consists of a group of protocols on their standard port numbers - FTP/FTPS on port 21, and SSH/SCP/SFTP on port 22 (and optionally HTTP on port 80 and HTTPS on port 443). These are the port numbers and protocols most often used, and generally will not need to be modified, although the port numbers can be changed if required. In the *Standard Edition* and *Professional Edition*, the default site is configured via the *Settings* panel.

**Admin site**

The "Admin site" is what the CompleteFTP manager connects to, to administer the server instance. By default it listens on port 14982 for FTP/FTPS, and port 14983 for SSH/SFTP. It does not matter which of these protocols the manager connects with to the Admin site. The Admin site cannot be deleted (the server could not be administered). Settings should be changed with care - if the port numbers are altered administrators must be informed of the changes.

In the *Standard Edition* and *Professional Edition*, the Admin site is configured via the Admin Access panel. In the *Enterprise Edition*, the Admin site is displayed in *Sites* by checking the 'Show admin site' checkbox at the bottom left of the *Sites* panel.

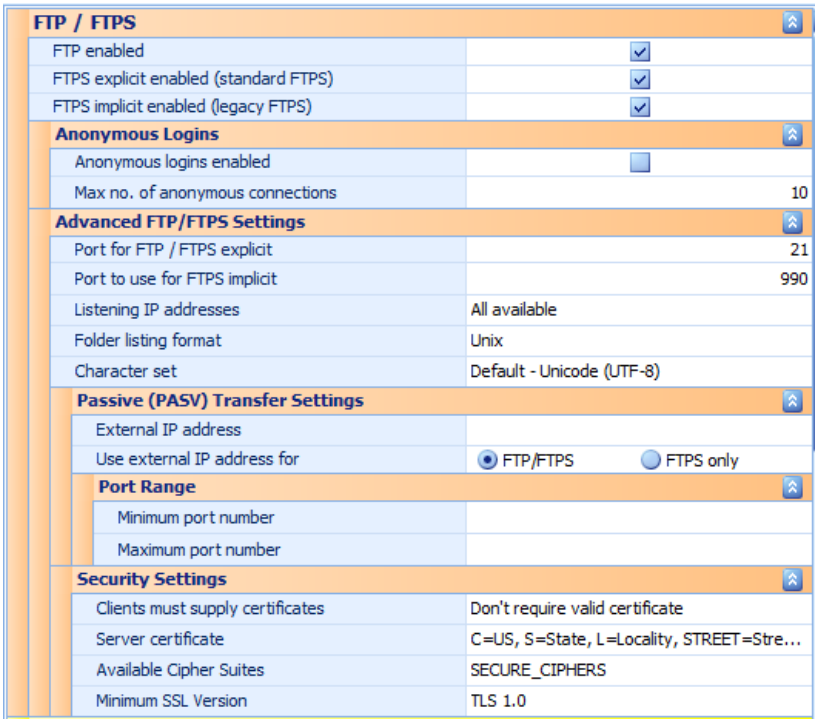
**Additional sites**

The *Enterprise Edition* of CompleteFTP supports the creation and administration of multiple sites.

# FTP/FTPS

FTP (File Transfer Protocol) is a well established Internet protocol designed to transfer files (and information about files) across networks using TCP (Transmission Control Protocol). FTP is defined in the Request For Comments 959 document (RFC 959), which can be obtained from the Internet Engineering Task Force.

FTPS is FTP over SSL. There are two incompatible variants of FTPS: explicit and implicit. FTPS explicit is the most recent and most widely used variant. FTPS implicit is older and is generally being phased out. FTP and FTPS explicit can coexist on the same port (usually 21), in fact FTPS explicit connections start out as FTP connections before being switched to SSL. FTPS implicit requires its own dedicated port (usually 990).




The various FTP/FTPS settings that are available to configure *CompleteFTP* are described below. Note that settings must be saved via the *Apply Changes* button, otherwise they will be lost.

| Setting | Description |
|---|---|
| FTP enabled | If checked, the FTP protocol is enabled. If not checked, the FTP protocol will not be supported by the server (although FTPS may be enabled, below). |
| FTPS explicit enabled (standard FTPS) | If checked, the FTPS protocol (explicit mode) is enabled. If not checked, FTPS explicit mode will not be supported by the server. If "FTP enabled" is not checked and FTPS (either implicit or explicit or both) is, then users are forced to use FTPS (i.e. they must connect securely). |
| FTPS implicit enabled (legacy FTPS) | If checked, the FTPS protocol (implicit mode) is enabled. This is a legacy form of FTPS that is still widely used. If not checked, FTPS implicit mode will not be supported by the server. If "FTP enabled" is not checked and FTPS (either implicit or explicit or both) is, then users are forced to use FTPS (i.e. they must connect securely)./td> |
| Anonymous logins enabled | If checked, anonymous access is enabled. This means connections can be made by logging in as an anonymous user rather than as a named user with a password. |
| Max no. of anonymous connections | The maximum number of simultaneous anoynmous connections permitted to the server. |
| **Advanced FTP/FTPS Settings** | |
| Port for FTP/FTPS explicit | Port that is used for FTP and FTPS in explicit mode. The standard port number for FTP and explicit FTPS is port 21. |
| Port for FTPS implicit | Port that is used for FTPS in implicit mode. The standard port number for implicit FTPS is port 990. |
| Listening IP addresses | Lists the IP addresses to listen on for this protocol. All network interfaces can be selected, or individual interfaces selected. |
| Folder listing format | Format of folder listings being sent back to clients. In the FTP protocol folder listings are sent back to the client as plain text. This setting controls the format of this plain text. This is particularly significant for non-console client applications, which usually parse the directory listings automatically. |

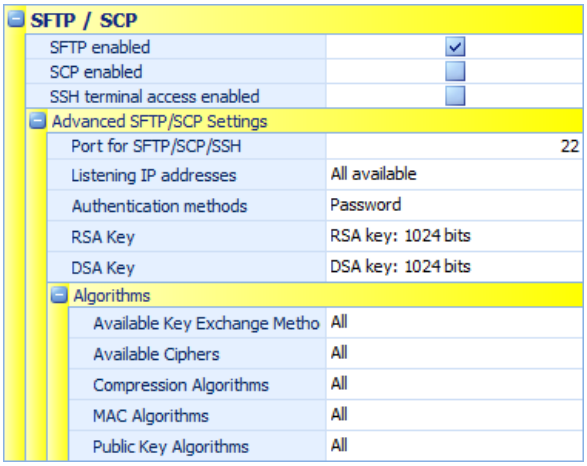
| | |
|---|---|
| Character set | Character set that is used for encoding file-names. |
| **Passive (PASV) Transfer Settings** | |
| External IP address | In passive (PASV) mode transfers, the server listens on a random port between the minimum and the maximum port numbers specified, waiting for a connection from the client to this port. The server sends the client a reply to the PASV command that provides the IP address to connect to and the port number. This setting allows a hard-coded IP address to be set. If not set, the IP address of the interface that the server is listening on is provided. This setting is required in cases where the IP address of the server is not reachable by clients. For example, the client may need to connect to another IP address that is accessible externally, and a NAT device may direct the connection to the server. |
| Use external IP address for | In passive mode transfers, some routers don't cope with a hard-coded external IP address as set in *External IP address* - they expect to see the local LAN IP address in the PASV reply, and they replace this local address with the external IP address themselves. If the local LAN IP address is not found, some routers will abruptly terminate the connection. The solution in these situations is to only supply the hard-coded external IP address for encrypted (i.e. FTPS) connections, which the router can't inspect (and hence can't know what IP address is in the PASV reply). To do this, select the "FTPS only" setting rather than "FTP/FTPS" (which is the default). |
| Minimum port number | This is the minimum port number used in passive (PASV) mode. In PASV mode transfers, the server listens on a random port between the minimum and the maximum port numbers specified, waiting for a connection from the client to this port. |
| Maximum port number | This is the maximum port number used in passive (PASV) mode. In PASV mode transfers, the server listens on a random port between the minimum and the maximum port numbers specified, waiting for a connection from the client to this port. |
| **Security Settings** | |
| Clients must supply certificates | This option controls whether or not certificates should be requested from FTPS clients. If they are required then the client must send a certificate that may be validated against the Windows certificate store. This means that either the client's certificate itself, or the certificate of one of its issuers must be in the Windows 'Trusted root certificate authorities' store. |
| Server certificate | Displays the dialog for controlling the server certificate. A server certificate must be set if FTPS or HTTPS is enabled. The dialog displays the current certificate's properties, and enables a new self-signed certificate to be generated. Also, a certificate signing request (CSR) can be created to send to a certificate authority (CA) to obtain a CA certficate.<br><br>This property is shared with the HTTP/HTTPS settings. |
| Available cipher suites | This setting controls the cipher suites that can be used. Generally, the default is appropriate unless there are special requirements for a particular algorithm. |
| Minimum SSL version | This setting controls the minimum version of SSL/TLS supported on this site (for both FTPS and HTTPS). The default is TLS 1.0, which means that SSL 3.0 is not supported by default. This ensures that the server is not vulnerable to the security flaws of SSL 3.0. |

## SFTP/SCP/SSH

SFTP is an abbreviation for "SSH File Transfer Protocol"", and is exactly that - a protocol for transfering files over an SSH connection. SFTP is not the standard FTP protocol running over SSH. Although SFTP has similar capabilities and even similar commands to standard FTP, these similarities are purely superficial. The protocol is completely different and incompatible with FTP and its secure extension, FTPS. SCP is also a file transfer protocol that runs over SSH connections. It is a precursor to SFTP, and allows the copying of files and directories over SSH.

The various SFTP/SCP/SSH settings that are available to configure *CompleteFTP* are described below. Note that settings must be saved via the *Apply* button, otherwise they will be lost.

Note that SCP and SSH terminals are not supported in the *Standard Edition* - the Professional or Enterprise Edition is required.

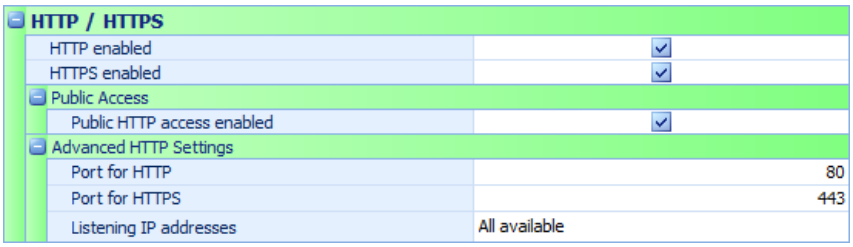



| Setting | Description |
|---|---|
| SFTP enabled | If checked, the SFTP protocol is enabled. If not checked, the SFTP protocol will not be supported by the server. |
| SCP enabled (not available in *Standard Edition*) | If checked, the SCP protocol is enabled. If not checked, the SCP protocol will not be supported by the server. |
| SSH terminal access enabled (not available in *Standard Edition*) | If checked, users may (if permitted) log into the server using an SSH terminal. If not checked, SSH terminal access will not be supported by the server. |
| **Advanced SFTP/SCP/SSH Settings** | |
| Port for SFTP/SCP/SSH | Port that is used for SFTP, SCP and SSH. The standard port number is port 22. All of these protocols use the same port number. |
| Listening IP addresses | Lists the IP addresses to listen on for these protocols. All network interfaces can be selected, or individual interfaces selected. |
| Authentication methods | Methods of user-authentication that the server accepts. Note that these methods can also be specified at the user level. When a user logis in, only the authentication methods specified at both levels are available. For example, if the site permits password only, and the user permits password and publickey, then only password will be available. Available methods are:<br>• *Password* - only a password is required. A password must be set for each user using the Users view.<br>• *PublicKey* - public key cryptography is used to authenticate the user. For every user the server must have a public key that matches the client's private key. Note that this method of authentication does not work for Windows users, since a password is required by Windows to log the user in. Thus *Password* or *PublicKeyAndPassword* should be enabled if Windows users are to be allowed to connect.<br>• *PublicKeyAndPassword* - both password and public-key authentication is used. |
| RSA key | The RSA server key - the private key that the server uses to identify itself. The server normally has a DSA and an RSA key set. This can be imported, exported or generated via the menu that appears when selecting the elipses (...) in the text box displaying the key's details. Note that generation of a new server key can take up to 30 seconds. The private key can only be imported or exported if the CompleteFTP manager is running on the server that the CompleteFTP service is running on. |

| | |
|---|---|
| DSA key | The DSA server key - the private key that the server uses to identify itself. The server normally has a DSA and an RSA key set. This can be imported, exported or generated via the menu that appears when selecting the elipses (...) in the text box displaying the key's details. Note that generation of a new server key can take up to 30 seconds. The private key can only be imported or exported if the CompleteFTP manager is running on the server that the CompleteFTP service is running on. |
| **Algorithms** | |
| Available key exchange methods | Controls what key exchange methods are supported by the server. It is rare that the defaults should be changed. It is possible that an administrator may want to disable weaker key exchange methods. |
| Available ciphers | Controls what ciphers are available to SSH for encrypting data. The default setting of All should generally be used unless some ciphers are required to be disabled. |
| Available ciphers | Controls what ciphers are available to SSH for encrypting data. The default setting of All should generally be used unless some ciphers are required to be disabled. |
| Compression algorithms | Permits the compression algorithms used in SSH to be selected. For example, if zlib compression is to be forced, then 'None' would need to be unselected. |
| MAC algorithms | Permits the MAC algorithms used in SSH to be selected. The default setting of All should generally be used unless some MACs are required to be disabled. |
| Public key algorithms | Controls what public key algorithms are supported by the server. For example, RSA public keys can be forced by disabling DSA. |

# HTTP/HTTPS settings

The various HTTP/HTTPS settings that are available to configure *CompleteFTP* sites are described below. Note that settings changes must be saved via the *Apply Changes* button, otherwise they will be lost.




Please note that HTTP and HTTPS should not be enabled until any other HTTP/HTTPS services on the server machine are shut down, e.g. IIS and Apache. Other products such as Skype also use the HTTP and HTTPS default ports of 80 and 443. If in doubt, use *netstat* to check what ports are in use. Note that these products can be run simultaneously, but must be configured so their port numbers do not clash. Both Apache and CompleteFTP can be set up with HTTP/S port numbers that differ from the default ports.

There are additional HTTP settings in *Limits & Timeouts*.

For HTTPS security settings, see the FTP/FTPS Security Settings (as they are shared).

| Setting | Description |
|---|---|
| HTTP Enabled | If checked, the HTTP protocol is enabled. If not checked, HTTP will not be supported by the server. If enabling, ensure that other HTTP servers are not running on the server machine. |
| HTTPS Enabled | If checked, the HTTPS protocol is enabled. If not checked, HTTPS will not be supported by the server. If enabling, ensure that other HTTPS servers are not running on the server machine. |
| Public HTTP access enabled | If checked, unauthenticated users will be able to access any file in the public folder. Unauthenticated users are represented by the user called 'anonymous'. By default the home-folder of this user is /Public, so this is the folder that will be publicly accessible if this option is enabled. |
| Server-side Javascript (JSS) enabled | Enable JSS for this site. Read more about JSS here. |
| Port for HTTP | Port that is used for HTTP. The standard port number for HTTP is port 80. |
| Port for HTTPS | Port that is used for HTTPS. The standard port number for HTTP is port 443. |
| Listening IP addresses | Lists the IP addresses to listen on for this protocol. All network interfaces can be selected, or individual interfaces selected. |
| Server certificate | Displays the dialog for controlling the server certificate. A server certificate must be set if FTPS or HTTPS is enabled. The dialog displays the current certificate's properties, and enables a new self-signed certificate to be generated. Also, a certificate signing request (CSR) can be created to send to a certificate authority (CA) to obtain a CA certficate.<br><br>This property is shared with the FTP/FTPS settings. |
| Use browser-based logins | If this option is enabled then HTTP logins to CompleteFTP will be handled by browser-based authentication (usually via basic authentication) rather than by the CompleteFTP login page. Note that basic authentication is not secure unless performed over SSL (i.e. using HTTPS). |
| Web features | Displays the configured web features. Clicking on the ellipsis (...) will bring up a dialogue box listing each web feature, and showing the configured URL path used by clients to access the application, and whether or not it is enabled. The URL path can be modified here, and the enabled or disabled status. By default, the Professional and Enterprise Editions support the FileManager feature that permits browsers to manage the remote file system. The *FileServer* application is responsible for serving files via HTTP. If it's disabled, individual files will be unavailable HTTP. The *FileSharing* feature controls access to shared files via HTTP. The *Login* feature performs HTTP user logins and the *CustomCommand* feature controls the Custom Command Script feature. |

| | |
|---|---|
| Listing template | Controls the format in which directory listings are displayed to the user. Directories may be listed by users by entering a URL specifying the path of a directory. The listing is rendered using HTML containing macros, such as %FileName%. Clicking on the ellipsis (...) will bring up a dialog permitting the user to enter their own template. See How to format HTML listings for more information. |
| Error template | Controls the format in which HTTP errors are displayed to the user. The default template uses Javascript to redirect the user to their home folder for 403 errors. Clicking on the ellipsis (...) will bring up a dialog permitting the user to enter their own template. |
| Custom MIME types | A MIME type, or Internet media type, is a two part identifier for file formats. In its most basic form, it defines the type and the subtype for a file extension. For example, HTML files are defined as "text/html". The webserver sends the MIME type to the browser as part of the file being downloaded, and the browser uses the MIME type to decide how best to display the file. An HTML file, with a MIME type of "text/html", will be displayed as a web page. Most common MIME types are included by default in CompleteFTP. Custom MIME types can be added, either as additional file types or to override default MIME types. |

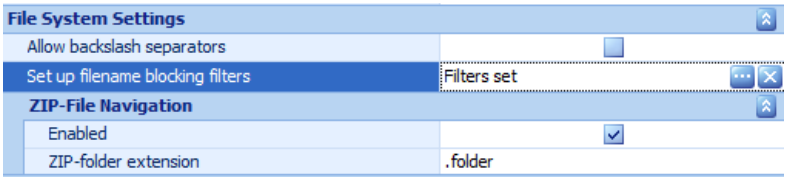
## File Sharing Settings

The file sharing settings that are available to configure *CompleteFTP* sites are described below. Note that settings must be saved via the *Apply Changes* button, otherwise they will be lost. There is also a per-user setting to enable or disable sharing.

| Setting | Description |
|---|---|
| Enabled | If checked, file-sharing is enabled. If not checked, file-sharing will not be supported by the server. Note that users are enabled for sharing by default. |

# Filesystem settings

The file system settings that are available to configure *CompleteFTP* sites are described below. Note that settings must be saved via the *Apply Changes* button, otherwise they will be lost.




| Setting | Description |
|---|---|
| Allow backslash separators | CompleteFTP uses Unix-style forward slash separators in paths. Occasionally it is desirable to be able to use Windows-style backslash separators - for example if there are clients accessing the server that only support backslash separators. CompleteFTP does not support backslash separators by default, but enabling this option enables support for backslash separators (forward slash separators are still supported if this option is enabled). |
| Set up filename blocking filters | Sometimes it is a requirement to block the upload of certain file types, such as executable files. This setting allows you to specify the filters used to control what files can be uploaded. Note that the filters are also applied to renaming of files - otherwise a user could upload a file with an acceptable name and then rename it to a banned name. Administrators can choose to block files that match the filters, or to only permit files that match the filters. |
| ZIP-file navigation enabled | When ZIP-file navigation is enabled ZIP files are presented to clients as though they are folders. This means that users can change into folders within the ZIP file and download specific files. This can save time and bandwidth as it allows users to download only what they need from a ZIP file instead of downloading the whole ZIP file. |
| ZIP-folder extension | By default, both the ZIP file itself and the folder that corresponds to the ZIP file are shown in listings, the latter having the additional extension '.folder'. The additional extension can be changed by modifying the 'ZIP-folder extension' setting. If this setting is cleared completely then the ZIP file won't be displayed being replaced by the corresponding folder. |

# General user settings

The general user settings that are available to configure *CompleteFTP* sites have been moved to the *Users* tab, and can be displayed by selecting the *General user settings* link. They are also described here.

# IP Filtering and Auto-Banning

CompleteFTP (Professional and Enterprise Editions only) supports *IP filtering* and *auto-banning* to better secure the server from attack. The Standard Edition also performs autobanning, but this feature is not configurable in the Standard Edition.

**IP filtering**

IP filtering allows rules to be established to prevent certain IP addresses from successfully connecting, or to allow only certain IP addresses to connect.

An IP filter consists of a collection of rules. Each rule consists of a *mask* and an *action*. The mask specifies which IP addresses the rule pertains to and the action specifies what should be done. Each rule also has a scope that is either the entire site or a specific user.

The mask may be (a) a specific IP address, such as 123.123.123.123, (b) the first one, two or three parts of an IP address, such as 192.168, which specifies all the IP addresses that begin with those parts, (c) a specific numeric range of IP addresses, such as 192.168.2.100-150, (d) an Internet host-name, or (e) a LAN network name.

There are three possible actions - *deny*, *allow* and *allow-always*. *Deny* will block access from clients whose IP addresses are included in the mask. *Allow* will permit access, unless the auto-banning system has detected too many failed login-attempts (see below). *Allow-always* is like allow except that auto-banning is disabled.

Rules are combined to filter incoming IP adddresses. When a connection attempt is made, the rules are consulted to see if the remote IP address matches the rules, and the connection is dropped if it is determined that access should be denied.

The *precedence* of operations is important. By default, the precedence is *Deny over allow.* This means that if all IP addresses are allowed, and there is a deny rule for a specific address or range, then the deny rule takes precedence and any IP address matching the deny rule is denied. *Allow over deny* is the other option, which means if an IP address matches an allow rule, then it will be permitted access even if it is listed in a deny rule.

Each rule may optionally be associated with a user. If it is associated with a user then it applies only to that user. If it's not associated with a user then it applies to the entire site. Typically *user-specific rules* are used in scenarios where all connections are rejected except those that are from specific users on specific IP addresses. To achieve this aim, create a rule that denies connections from all IP addresses and set the precedence to 'allow over deny', then add user-specific rules. User-specific rules are most easily created via the 'Allowed IP addresses' property in the User Properties panel. When this property is set an 'allow rule' is created for the specified IP addresses for that user on each site on which the user is enabled.

**Auto-banning**

*Auto-banning* is an important feature related to IP filtering that helps secure the server. It works by automatically banning IP addresses from connecting (for a period of time) if they have failed to authenticate a certain number of times within a time period.

For example, using the default settings, if a rogue client is attacking from a given IP address, and fails to guess a password correctly 10 times within a 60 second period, their IP address is banned from connecting for the next hour. After an hour has elapsed, the ban is automatically lifted.

The auto-banning settings are configurable, although it is generally advisable to use the defaults.

The currently banned IP addresses can be viewed in the *Monitoring* panel. Bans can be removed or made permanent from this panel.

Note that auto-bans are not persistent - if the service is restarted, all bans are lost (unless they are made permanent via the *Monitoring* panel).

Auto-banning can be disabled for specified IP addresses using an allow-always IP filter-rule (see above). Clients connecting from IP addresses that fall under an IP filter-rule whose action is *allow-always* will be immune from auto-banning. This is especially helpful in situations where many clients are connecting from a single Internet IP address since it stops a single 'misbehaving' client from causing all other clients to lose their ability to connect.

## Limits and Timeouts

The various limits and timeouts that are available to configure *CompleteFTP* sites are described below. Note that settings must be saved via the *Apply Changes* button, otherwise they will be lost.

| Setting | Description |
|---|---|
| Timeout for HTTP sessions | The maximum time in seconds before HTTP sessions expire. If the timeout is set to zero, sessions do not expire. The default is 1,800 seconds (30 minutes). |
| Timeout for logging in | The maximum time in seconds permitted for a login attempt. If a connection is made and login is not completed within this period, the connection is closed. If the timeout is set to zero, the timeout is infinite, i.e. the connection does not time out. |
| Timeout for stalled transfers | The maximum time in seconds that the underlying sockets wait for while performing read operations. If the timeout is set to zero, the timeout is infinite, i.e. read operations do not time out. |
| Timeout for idle sessions | The maximum time in seconds that a connection is permitted to be idle, i.e. if no commands are sent from the client. If the timeout is set to zero, the timeout is infinite, i.e. the connection does not time out. |
| Timeout for PASV wait | In passive mode, the server listens on a socket, waiting for the client to make a connection. The client can connect before actually sending the next command (LIST,NLST,RETR or STOR), and the server must wait for this next command before it can do anything with the client connection. If the timeout is set to zero, the timeout is infinite, i.e. the connection does not time out. |
| Max. allow login attempts | Sets the maximum number of login attempts that can be made for a user account. |
| Max. simultaneous connections | Sets the maximum number of simultaneous connections permitted. If this limit has been reached, any further connection attempts are denied until current connections are closed. This setting does not apply to HTTP connections. |
| Max. simultaneous HTTP connections | Sets the maximum number of simultaneous HTTP connections permitted. If this limit has been reached, any further HTTP connection attempts are denied until current connections are closed. |
| Max. connections per user | Sets the maximum number of simultaneous connections permitted per user. If this limit has been reached for a user, any further connection attempts by that user are denied until current connections are closed. |

# Messages

The message settings that are available to configure *CompleteFTP* sites are described below. Note that settings must be saved via the *Apply Changes* button, otherwise they will be lost.

| Setting | Description |
|---|---|
| Welcome message | The text that is displayed (i.e. sent to the client) during a login attempt. In SFTP this is the banner message, and is often not displayed by SFTP clients. |
| Hide server product details | If this option is enabled, the product name and version of the server is not displayed in welcome messages. This may be required for PCI compliance. This option is not enabled by default. |

# How to secure your server

FTP servers are always vulnerable to attack from unauthorised people, and a number of steps should be taken to minimize the risks of this occurring:

1. Keep up to date with the latest patches for your operating system. Microsoft regularly publishes security updates, so ensure that they are applied to your host.
2. Keep up to date with the latest release of CompleteFTP. Each release usually includes at least one security fix, and it is important to keep your installation current.
3. A good firewall is the first line of defence for security, and should be your first port of call in denying/permitting access to certain IP addresses.
4. If your server is not required to be accessible from the Internet, ensure that it is only reachable internally. If it is not accessible externally, the only attacks can be from within your organization, greatly reducing the risk.
5. Use the IP filtering capabilities of CompleteFTP to only permit the IP addresses you want (if this is possible).
6. Ensure that auto-banning is configured correctly (e.g. the defaults) to prevent dictionary attacks on passwords.
7. Regularly review log files for unwanted intrusions and take remedial action (such as banning IP addresses).
8. Disable protocols that aren't being used, e.g. if you are running an SFTP server only, disable FTP, FTPS, HTTP, HTTPS and SCP.
9. Don't use HTTP - use HTTPS. Usernames, passwords and other sensitive data should not be sent via HTTP.
10. Don't use FTP - use FTPS. Usernames, passwords and other sensitive data should not be sent via FTP.
11. Disable SSL 3.0. SSL 3.0 is disabled by default starting from CompleteFTP 8.1.3. SSL 3.0 has serious security flaws that make it unsafe for use, e.g. the POODLE attack
12. Disable the automatic Windows users feature, so that only explicitly permitted users are permitted.
13. Use strong password policies - see here. Also disable any anonymous FTP access unless it is required.
14. For SFTP disable password authentication, and only permit public key authentication. This means users must have valid private keys and have their public keys registered on the server. This is not always possible of course.
15. For SFTP disable SSH terminal access. This is disabled by default. SSH terminal access permits Windows users who have this feature enabled to execute almost any program or DOS command, and is potentially a significant security hole.
16. Hide the server product details. This is found in the Settings->Messages section, and conceals the version and name of the product from clients. This is disabled by default. The default welcome message should also be changed, as it identifies the server.
17. Enforce hashed passwords for database users. If database users are enabled (not the default), make sure any stored passwords are hashed. Even better, use a 'salt' prepended to passwords before they are hashed. See *how to configure database users*

# How to set a default Windows domain

If a default Windows domain is set, it means that Windows users on the default domain can login without specifying the domain (instead of using MyDomain\MyUserName they can use MyUserName). This can be very convenient for users.

The default Windows domain tells CompleteFTP which domain to authenticate against if (1) a domain-name was not given and (2) no exact match was found in CompleteFTP's list of users.

For Windows users that have been explicitly added to CompleteFTP, if the client tries to log in using a username without a domain-name then the server will first look for this exact username in the config (just like it does now). If a match is not found and a default domain has been set then it tries to prefix the default domain to the username and look up the user-list.

If the user is not found in the explicitly added list of users and automatic Windows users are enabled, the default domain is again used if the domain name was not specified while logging in. This will result in logging in as 'DefaultDomain\MyUserName'. One consequence of this is that if a default Windows domain has been set then an attempt to log in without specifying a domain name will never log into the local machine. CompleteFTP will always assume the default domain should be used. If the user does want to log into the local machine when a default domain has been set they have to log in using ".\MyUserName".

# How to use HTTP/HTTPS

CompleteFTP supports HTTP and HTTPS in the *Professional* and *Enterprise* Editions.

HTTP and HTTPS may be enabled by selecting the "HTTP enabled" and "HTTPS enabled" settings (see the HTTP/HTTPS group of settings).

Please note that HTTP and HTTPS should not be enabled until any other HTTP/HTTPS services on the server machine are shut down, e.g. IIS, Apache. Other products such as Skype also use the HTTP and HTTPS default ports of 80 and 443. If in doubt, use *netstat* to check what ports are in use. Note that these products can be run simultaneously, but must be configured so their port numbers do not clash. Both Apache and CompleteFTP can be set up with HTTP/S port numbers that differ from the default ports.

How to use HTTP

How to use HTTPS

How to perform HTTP uploads

# How to use the File Manager

The Professional and Enterprise Editions of CompleteFTP support HTTP and HTTPS. Since version 7.2, they both support the FileManager web application - a Javascript/HTML client that can be used in any browser to manage the remote file system. This is an extremely powerful GUI web client that can perform all the normal operations expected of a file manager application, including uploading and downloading files. An example is shown at the bottom of this page.

**Enabling the FileManager**

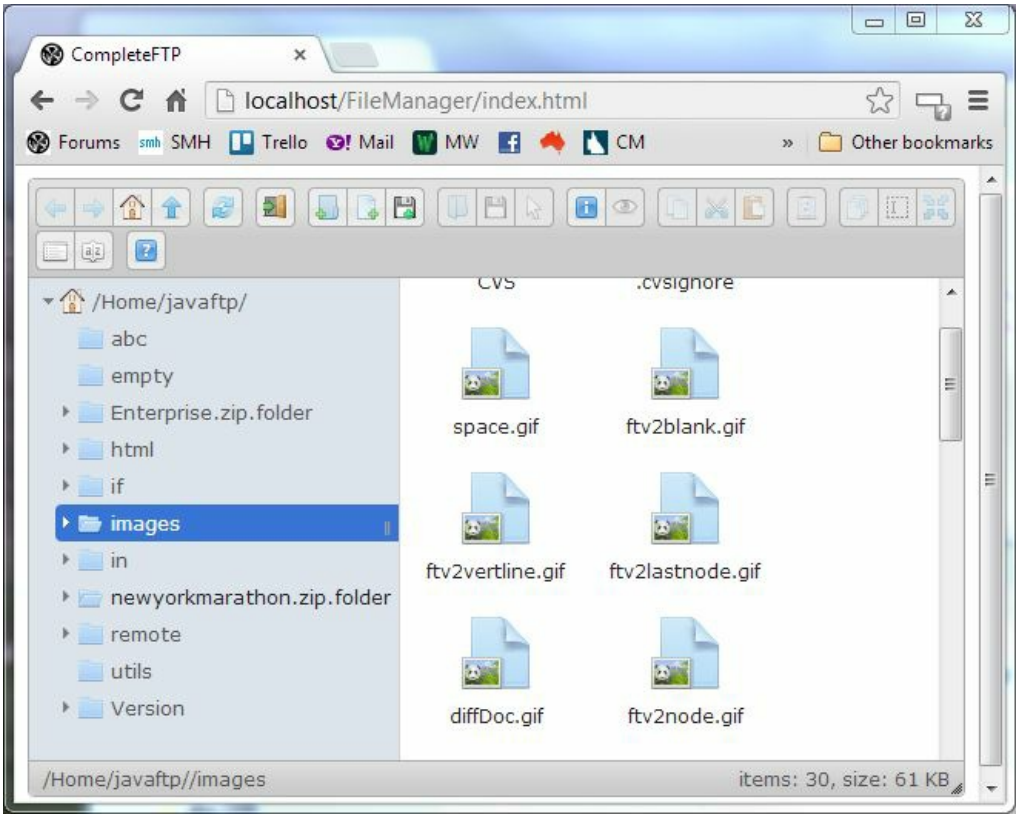
To enable the FileManager web-app, open the CompleteFTP manager and go to the Sites tab. Go down to the HTTP/HTTPS section. Ensure that HTTP (and/or HTTPS) is enabled, and go to the Advanced HTTP Settings. Select the Web-apps ellipsis (...). More details can be found here.

**Accessing the FileManager**

By default, the FileManager (if enabled) is available at the /FileManager URL on the server, but users are also redirected there when they access their home directories. If users have not authenticated, they will be prompted to do so by the standard browser authentication dialog. Once authenticated, the FileManager will be displayed. It functions like most file managers. There's a right hand click menu on both folders and files that can be used to perform various actions. Files can also be dragged and dropped onto the FileManager and they will be uploaded.

**Customizing the FileManager**

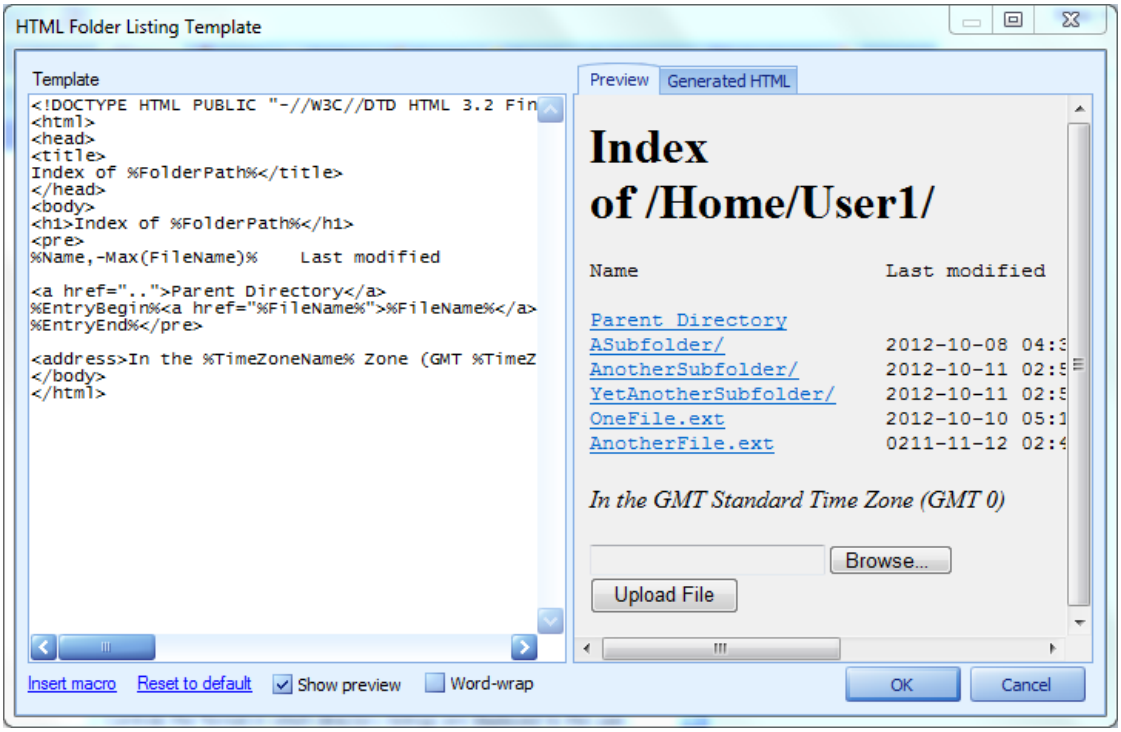
The FileManager is based on the elFinder file manager, and is written entirely in Javascript. Accordingly, it can be modified (at your own risk!). The code can be found at *C:\ProgramData\Enterprise Distributed Technologies\Complete FTP\FileManager* or the equivalent (depending on your OS). The elFinder website contains documentation, including client configuration options. Different languages are also supported.



# How to format HTML listings

When a HTTP client requests a folder URL that it has permission to list, CompleteFTP will return a listing in HTML. The format of this listing is controlled by a template that may be modified. The template is defined by a single HTML page containing macros (see the 'Template Format' section below).

The template is modified using an editor that features real-time previews. Clicking on the ellipsis (...) in the Listing template property displays this editor, shown below.




The HTML to be used for the folder listing is entered on the left. On the right are shown the rendered HTML, and the generated HTML (i.e. with macros expanded). Macros may be entered manually or inserted by clicking the 'Insert macro' link at the bottom. The 'Reset to default' link may be clicked to set the template back to CompleteFTP's default format.

## Template Format

The template is split into three sections: the header, the entry-block and the footer. The header is everything before the `%EntryBegin%` macro; the entry-block is everything between `%EntryBegin%` and `%EntryEnd%`; and the footer is everything after `%EntryEnd%`. When a folder-listing is rendered the entry-block is repeated once for each entry in the folder. An entry can be a file or a sub-folder.

The table below defines the available macros as well as stating which section each macro may be used in:

| Macro | Section | Description |
|---|---|---|
| `%EntryBegin%` | Start of entry-block | Marks the start of a block that defines the format of a single entry in the listing - file or folder |
| `%EntryEnd%` | End of entry-block | Marks the end of a block that defines the format of a single entry in the listing - file or folder |
| `%FileCreated%` | Entry-block | Date and time at which the file or folder was created |
| `%FileGroup%` | Entry-block | Group of the folder or file |
| `%FileLength%` | Entry-block | Length of the file in bytes, kB, MB or GB |
| `%FileModified%` | Entry-block | Date and time at which the file or folder was last modified |
| `%FileName%` | Entry-block | Name of the file or folder (not including the path) |
| `%FileOwner%` | Entry-block | Owner (user) of the folder or file |
| `%FilePath%` | Entry-block | Full virtual file-system path of file or folder |
| `%FilePermissions%` | Entry-block | UNIX-like permission string of the file or folder |
| `%FileType%` | Entry-block | Type of the directory-entry - either 'File' or 'Folder' (useful for CSS classes) |
| `%FolderName%` | Header or footer | Name of the folder being listed |
| `%FolderPath%` | Header or footer | Full virtual file-system path of the folder being listed |
| `%TimeZoneName%` | Header or footer | Name of the current time-zone |
| `%TimeZoneOffset%` | Header or footer | Offset (in hours) of the current time-zone compared to UTC |

**Formatting:**
Formatting instructions may be provided with each macro to refine how it is rendered. These instructions use a syntax similar to the composite formatting syntax defined by Microsoft. The general pattern is:

```
%macro-name[,alignment][:formatString]%
```

where `macro-name` is the name of the macro as shown in the table above or plain text (see below); `alignment` is an integer defining the preferred width of the field (positive=left-aligned, negative=right-aligned); and `formatString` provides instructions on how the text is formatted. These instructions depend on whether the field's type is a date or a number.

The functions, `Max(m)` and `Pad(m)`, may be placed in a format-string. `m` is the name of an entry-block macro, such as

`FileName`. `Max(m)` returns the maximum width of the field, `m`. `Pad(m)` returns the difference between the maximum and minimum widths of the field, `m`.

If the text that is not a legal macro-name is provided then that text is rendered literally. If no macro-name is provided then spaces will be inserted according to the `alignment` format. For example, `,Max(FileName)` will insert a number of spaces equal to the longest file-name.

**Example:**

The following example produces a listing similar to the default listing format of Apache:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
<title>Index of %FolderPath%</title>
</head>
<body>
<h1>Index of %FolderPath%</h1>
<pre>
%Name,-Max(FileName)%    Last modified            Size      Description

<a href="..">Parent Directory</a>
%EntryBegin%<a href="%FileName%">%FileName%</a>%,Pad(FileName)%    %FileModified,-25:yy
%EntryEnd%</pre>

<address>In the %TimeZoneName% Zone (GMT %TimeZoneOffset:+#;-#;0%)</address><FORM METHO
</body>
</html>
```

***Notes:***

1. Since `Name` is not a legal macro-name, the text `%Name,-Max(FileName)%` will render the literal characters `"Name"` followed by the number of spaces required to yield the same total width as the longest file-name.
2. `%,Pad(FileName)%` yields a number of spaces equal to the difference between the longest and shortest file-names.
3. `%FileModified,-25:yyyy-MM-dd hh:mm%` renders a file's last-modified date left-aligned in the format "2012-15-11 14:02" and padded with spaces to give a total width of 25 characters.
4. `%TimeZoneOffset:+#;-#;0%` yields the timezone offset hours with a plus- or minus-sign in from of it.

## How to customize the welcome page

When a user first navigates to the server via a web browser, a default HTML page is displayed. This page is easily modified. Depending on the CompleteFTP installation (and the operating system), this HTML page and associated files can be found at:

*C:\ProgramData\Enterprise Distributed Technologies\Complete FTP\Public\index.html*.

This page can be replaced with your own company-branded HTML page or pages.

# How to use CompleteFTP with firewalls

Firewalls block connections to networks (and thus servers) on certain port numbers, and can thus prevent client applications connecting.

If problems are being experienced connecting to CompleteFTP, the firewall is a good place to start looking.

HTTP and HTTPS rarely have firewall problems, while SFTP has fewer problems with firewalls than FTPS. All commands and transfers go through a single port (the default is port 22). As long as the firewall is permitting connections to port 22, SFTP should work correctly.

The situation is more complicated with FTP, mainly because directory listings and file transfers are performed using separate socket connections on a different (and potentially random) port number to the standard port 21. Of course firewalls are designed to prevent connections on random port numbers, so listings and transfers can get blocked.

From an FTP client perspective, the easiest way to enable FTP clients to function through server firewalls is for them to use passive mode transfers, also known as PASV. The client initiates all transfers, and the server returns an IP address and port number to connect to. The server is pre-configured to return port numbers within a certain range of ports that are set up for FTP access in the firewall.

To set up a passive (or PASV) port range with CompleteFTP, see how to set a PASV port range. This port range should also be set in the firewall, permitting clients to connect in this range.

Often, the FTP server will be on a machine with only an internal IP address. As the IP address is returned to the client via the PASV command, this is problematic. The client requires an external IP address, and the firewall must be configured to reroute packets to this external IP address to the server machine. To set an external IP address with CompleteFTP, see how to set an external IP address.

# How to set an external IP address

In passive (PASV) mode transfers (FTP and FTPS only), the server listens on a random port between the minimum and the maximum port numbers specified, waiting for a connection from the client to this port.

The server sends the client a reply to the PASV command that provides the IP address to connect to and the port number.

If the *External passive IP address* setting is set, the server sends back that IP address in the PASV reply. Otherwise, the IP address of the interface that the server is listening on is provided. This setting is part of the Passive (PASV) Transfer Settings group of settings.

Using the hard-coded external address is often necessary if the server is behind a NAT router or firewall, as the server may only know about its internal IP address. In this situation, the internal IP address can't be passed back to the client in the PASV reply - an external client can't connect to an internal IP address such as 192.169.x.y.

**Using the external IP address for FTPS only**

There is one caveat to using a hard-coded external IP address - some routers don't like it. These routers expect to see the local LAN IP address in the PASV reply, and they replace this local address with the external IP address themselves. If the local LAN IP address is not found, some routers will abruptly terminate the connection.

The solution in these situations is to only supply the hard-coded external IP address for encrypted (i.e. FTPS) connections, which the router can't inspect (and hence can't know what IP address is in the PASV reply). To do this, use the *Use external IP address for* setting and select "FTPS only" rather than "FTP/FTPS" (which is the default).

# How to set a PASV port range

In passive (PASV) mode FTP transfers (not applicable to SFTP), the server listens on a random port between the minimum and the maximum port numbers specified (the passive port range), waiting for a connection from the client to this port.

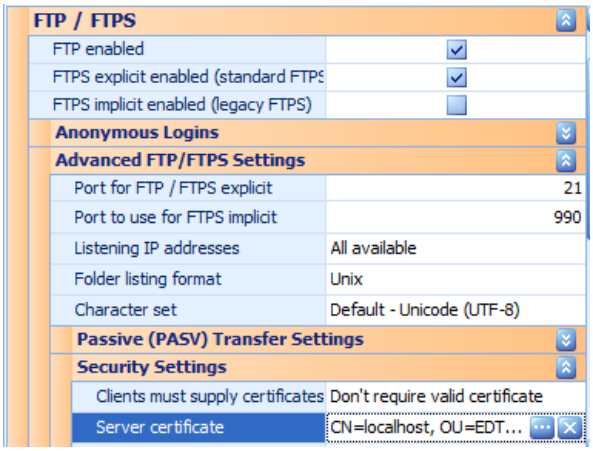
The minimum and maximum port numbers are set using the *Minimum port number* setting and the *Maximum port number* setting. These settings are part of the Passive (PASV) Transfer Settings group of settings. It is advisable to set a passive port range of at least 100.

If these values are not set explicitly, the defaults are used.

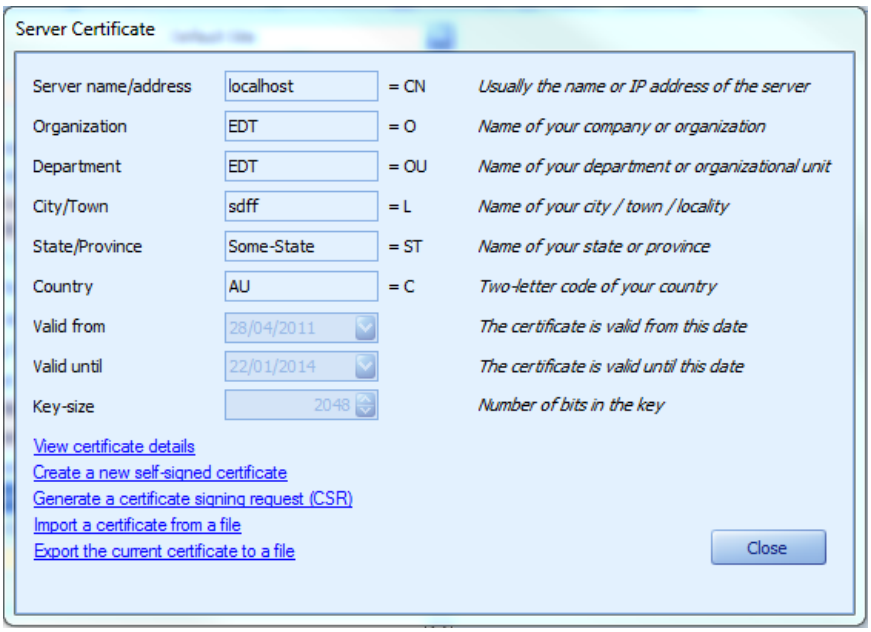
# How to set up the server certificate

When using FTPS and HTTPS, the server **must** have an SSL certificate installed. The certificate is an integral part of how the secure connection is established.

By default, a server certificate is already installed. It can be viewed via the *Server certificate* setting in the FTP/FTPS group of settings. Selecting the "..." button (see below) will bring up the Server Certificate dialog box.




The default certificate has a Common Name (CN) of "localhost", and is a self-signed certificate. By convention, the CN is often set to the hostname of the server for use in server validation, and another self-signed certificate with the desired hostname as CN can be generated by selecting "Create a new self-signed certificate" (below). This will replace the existing certificate.




On production machines, certificates issued by a certificate authority (CA) may be required. If the server will be accessed from HTTPS or FTPS clients external to the organization (e.g. across the Internet) then a CA certificate is essential. Many organizations already have CA certificates issued and centrally controlled. Otherwise, you will need to purchase and install a CA certificate.

# How to set up a CA SSL certificate

On production machines, the default self-signed SSL certificate will probably not be sufficient. Instead, a certificate issued by a certificate authority (CA) may be required. If the server will be accessed from HTTPS or FTPS clients external to the organization (e.g. across the Internet) then a CA certificate is essential. Many organizations already have CA certificates issued and centrally controlled. Otherwise, a CA certificate must be purchased and installed, as described below.

There are three steps in obtaining and installing a CA certificate:

1. Generate a Certificate Signing Request (CSR)
2. Purchase the CA certificate using your CSR
3. Download and install your CA certificate

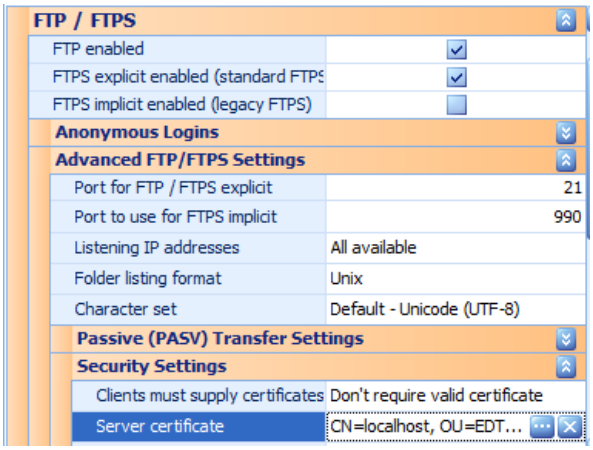
# How to generate a CSR

To purchase a CA SSL certificate from a CA, a Certificate Signing Request (CSR) must be generated. A CSR is a message sent to a CA that contains the information identifying the applicant, as well as the applicant's public key. This information (as well as the public key) is embedded in the issued certificate. It includes fields such as the organization name, city, region and country.

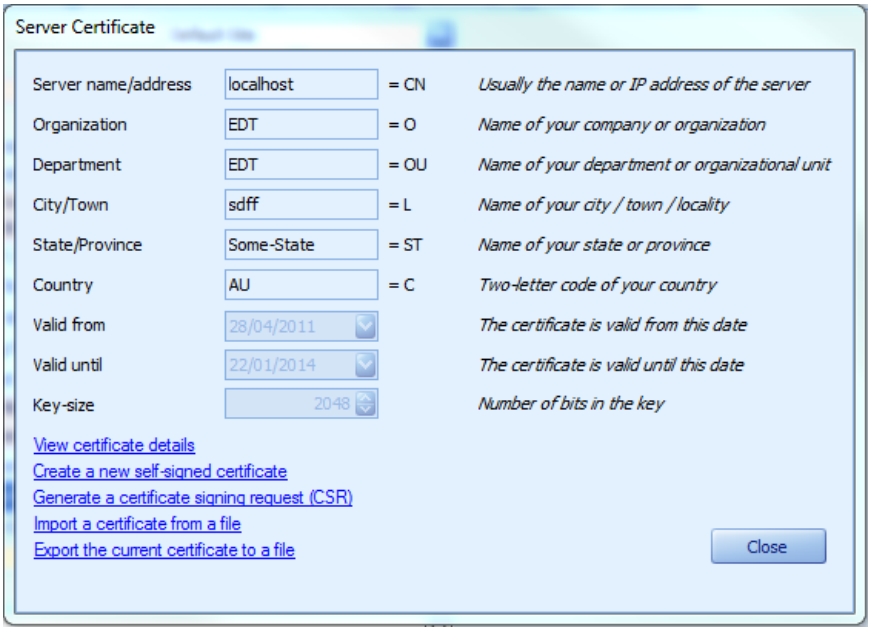
It is important to note that SHA-1 signed certificates are being phased out by CAs, and will not be trusted after January 1, 2017. The stronger SHA-2 family of hash algorithms is replacing SHA-1. As of version 8.1.6, CompleteFTP generates CSRs with SHA-2 (specifically SHA-256). Many CAs will no longer accept CSRs generated by earlier versions of CompleteFTP.

The most important field is the **Common Name (CN)**, which is normally the fully qualified domain name that you wish to secure using the certificate, e.g. 'www.mydomain.com' or 'myserver.mydomain.com'. Before the CSR is sent, it must be signed by the applicant's private key. The CA sends back the certificate containing these fields, signed by the public key of the CA.

The CompleteFTP manager can generate the keypair (private and public keys) required together with the CSR to be sent to the CA. In the Site/Sites panel, the FTP/FTPS group of settings must be chosen. Drill down to Advanced FTP/FTPS Settings and then to Security Settings. Click on the *Server certificate* field to display the ellipsis (...) button (see below) and select that to display the Server Certificate dialog box.




Select the *Generate a certificate signing request (CSR)* link (see image below) to begin the process of generating a CSR. Note that a private key (in PVK format) is first generated - this key must be saved to a secure location for use with the issued certificate. Do not lose it! Your certificate is of no use without the corresponding private key. The end result of the process should be a CSR file and a private key. The CSR file is to be sent to the CA you have selected. The private key is not sent to the CA.



## How to purchase a CA SSL certificate

Once you have your CSR (and have your CSR's private key safely stored), you can purchase an SSL certificate from a CA.

Firstly, you must select a CA to purchase from. There are a number of CAs that sell SSL certificates, and for compatibility reasons, it is best to choose one of the more well-known providers. Some of the largest are listed here. Please note that prices vary considerably, so compare prices before purchasing.

Purchasing will involve uploading your CSR to the CA's website. The CA will then want to verify your organization's identity before issuing the certificate. In particular, they will need to verify that your organization is the legal owner of the domain being set as the Common Name. They may require certain documents to be sent to them as part of the verification process, or send an email to the registered owner of the domain (to confirm you have access to this email).

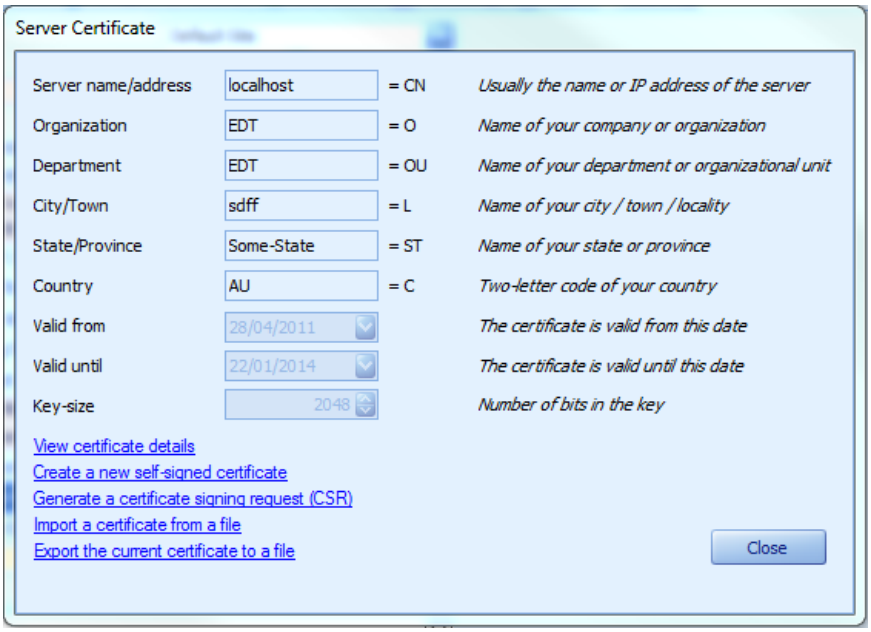
# How to install a CA SSL certificate

Your chosen CA will send you an email with a link to download your new certificate once their verification procedure is completed.

Most CAs will allow you to choose a format in which you can download your certificate. Generally, "Apache/SSL" or "Other" will be suitable for CompleteFTP - this will usually be a CER file that can be directly imported.

To import into CompleteFTP, use the *Import a certificate from a file* link in the *Server Certificate* dialog box (see below). The private key that was generated with the CSR will also be required - you will be prompted to select it.

If CompleteFTP was not used to generate a CSR, you may be provided with an SSL certificate and private key in different formats. See how to import SSL certificates in other formats.



# How to import SSL certificates in other formats

If CompleteFTP is used to generate a CSR so that a certificate can be requested from a certificate authority (CA), then the private key that is generated along with the CSR is stored in the Microsoft PVK format. The SSL certificate is normally supplied by the CA in CRT or CER format, which can usually be imported straight into CompleteFTP along with the PVK file (which is prompted for after loading the certificate file).

If CompleteFTP is not used to generate a CSR, your SSL certificate and its private key might be in other formats. The commonest private key format is PEM-encoded, and it will start with the line "-----BEGIN RSA PRIVATE KEY-----".

CompleteFTP cannot directly import SSL certificates with PEM-encoded private keys, but it is easy to convert them to a supported format using OpenSSL. The supported format is PFX, also known as PKCS#12 format. Windows binaries of OpenSSL can be obtained here. Normally one of the Win64 "Light" distributions is sufficient if you are running Windows 7 or Windows 8, or another 64 bit Windows OS.

**Please note** that OpenSSL should be run in a command prompt that is being run with administrator privileges.

Sometimes, your PEM-encoded certificate file will have the certificate authority's certificates embedded in it as well as your SSL certificate. Your certificate can usually be identified by the attributes listed above it (especially its common name).To import into PFX format (below), create a file that contains only your certificate (cut and paste inclusive from the "-----BEGIN CERTIFICATE-----" to "-----END CERTIFICATE-----").

Use the following OpenSSL command to convert your certificate and private key into PFX format:

```
openssl pkcs12 -export -out certificate.pfx -inkey privateKey.key -in certificate.crt
```

You can then import the PFX file directly into CompleteFTP via the *Import a certificate from a file* link in the *Server Certificate* dialog box.

**Converting a private key to PVK format**

Sometimes the certificate is already in PFX format, but the private key is PEM-encoded. In this situation, the private key can be converted to the Microsoft PVK format. When the PFX file is imported without a private key, the private key in PVK format is then prompted for. To do so, the pvk tool can be used as shown below:

```
pvk -in privateKey.key -out privateKey.pvk -topvk
```

Some users have found it easier to also convert their PFX certificate to CER format (as well as their private key into PVK format).

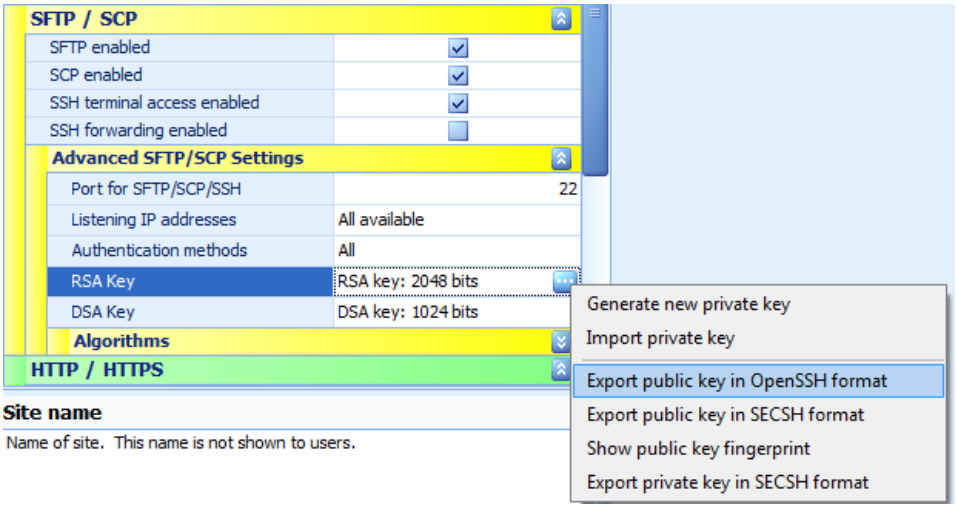
# How to set up the server keys

When using SFTP, the server **must** have private keys installed. The keys are an integral part of how the secure connection is established. They identify the server to its users.

There are two types of keys - DSA keys and RSA keys. When clients connect, they indicate what type they prefer to be used, and a key type is negotiated. In Settings->SFTP->SSH Settings->Public Key Algorithms, one or the other type of key may be disabled if, for example, only RSA is to be supported.

By default, private keys are already installed - they are generated when CompleteFTP is installed.

The private keys can be regenerated or private keys can be imported via the popup menu shown by selecting the ellipsis (...) on RSA Key or DSA Key, found via Settings->SFTP/SCP->Advanced SFTP/SCP Settings, as shown below. The server's public keys can also be exported via this menu.



# How to use SSH terminal access

CompleteFTP (Professional and Enterprise Editions only) supports SSH terminal access. This means that a command-line SSH client can SSH onto the CompleteFTP server machine, obtain a shell and use various command-line commands.

Note that CompleteFTP must have SSH terminal access enabled (disabled by default) and the user must also have SSH terminal access enabled. Both Windows and non-Windows (internal) users can be granted terminal access. However as non-Windows users run by default as the local SYSTEM user, for security reasons they are not permitted to exec processes.

Once the server settings (in SFTP/SCP/SSH settings) are enabled for SSH terminal access, users can login to CompleteFTP via *ssh username@hostname*, from any command-line SSH client on any operating system.

Once logged in, users are presented with a simple Unix-like command-line shell that permits certain commands, such as changing directory, creating directories, and copying and renaming files.

The *help* command lists all available commands.

**Exec command**

The most useful command (and only permitted for Windows users) is *exec*, which can be used to launch programs and scripts. For example, to run a DOS command such as 'echo', you would use the standard 'cmd' program, e.g.

*exec cmd /c echo test*

The same technique can be used to run the DOS 'type' command to display the contents of a text file:

*exec cmd /c type test1.txt*

For DOS commands, the '/c' flag is necessary as it instructs the DOS shell to terminate after running the command.

# How to use SSH tunneling

CompleteFTP (Professional and Enterprise Editions only) supports local SSH port forwarding (often known as SSH tunneling). This means SSH tunnels can be established between a client machine and CompleteFTP that other protocols can use.

Why would you use SSH port forwarding (also called SSH tunneling)? There are two reasons - firewalls and security.

Consider the scenario where an employee using a laptop outside the corporate network wants to give a demonstration to a customer using a machine within the corporate network. Perhaps they want to use RDP (the Remote Desktop Protocol) and it has not been set up in secure mode. In this scenario a CompleteFTP server is also within the corporate firewall.

**Firewalls**

By default, RDP uses port 3389, but say for security reasons the corporate firewall does not permit external connections to this port number. If SSH port forwarding is used, all RDP traffic is transmitted over port 22 (the standard SSH port). There is no need to open the RDP port in the firewall.

**Security**

What's more, the RDP protocol is transmitted securely over the SSH tunnel. So unsecure protocols can be safely used via SSH tunnels. The only unsecure portion of the route is between CompleteFTP and the destination machine (the RDP machine), and both of these machines are inside the corporate firewall.

**Setting up tunneling**

How is port forwarding set up? For security reasons, it is disabled by default in CompleteFTP. It must be enabled both for the site and individual users. To enable port forwarding for the site, the Site tab in the CompleteFTP manager must be opened, and SSH Port Forwarding flag enabled, which is under the SFTP/SSH settings section. The users who require port forwarding must also have this option enabled, which is done in the User panel by selecting the user.

On the client side, the tunnel must be set up by an SSH utility such as PuTTY. When PuTTY sets up a local port forward, it listens on the client for connections on the local port specified. When a connection on the client is made to this *local* port on the local machine, the SSH tunnel is established with CompleteFTP. All the data on this local connection is sent through the tunnel to CompleteFTP, which forwards it to the ultimate destination (set by PuTTY).

# How to configure anonymous logins

FTP servers often permit anonymous logins. This means users log in as the 'anonymous' user, and don't require a password.

In CompleteFTP anonymous logins are disabled by default, but can be enabled via the Settings window. The *Anonymous Logins* section has a checkbox for *Anonymous logins enabled*. This should be checked if anonymous logins are to be permitted.

To prevent malicious users logging in multiple times anonymously, the maximum number of anonymous logins can also be set. The default is unlimited, but if anonymous logins are enabled this should be set to a number in keeping with the site requirements.

# How to use multiple sites

Every installation of CompleteFTP supports at least two *sites* - the 'Admin' site and the default site. A site consists of a related group of protocols and port numbers. Sites have their own settings, including public keys and certificate.

The *Enterprise Edition* of CompleteFTP supports the creation and administration of additional sites, as described here.

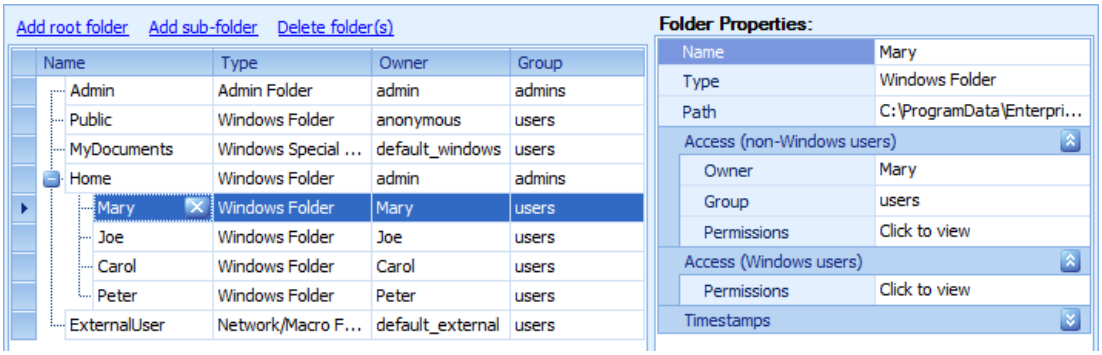
# Filesystem

CompleteFTP has an extremely flexible virtual file system, which means that folders can be mounted from almost anywhere on the physical file system (or network) and combined in various ways.

There are four different types of folders supported:

- *Virtual folders* don't exist on the actual file system - they permit a virtual folder structure to be created.
- *Windows folders* are standard Windows folders
- *Windows special folders* are folders such as *My Documents*.
- *Network/Macro folders* are UNC (network) folders, or folders with an expanded macro.
- *Custom folder-types* may be developed in the Enterprise Edition (see here).

CompleteFTP may have many root level folders mounted, and if virtual folders are used the CompleteFTP file system may not bear any resemblance to the underlying file system.




If the Hidden checkbox is checked then the folder will not show up in listings. This works independently of access permissions. For example, a folder may be shown but not accessible, or it may be hidden but still accessible.

Note that macros may be inserted into folder-names using the same %xyz% notation used elsewhere.

# How to create a virtual folder

Virtual folders don't physically exist. They are used to create virtual hierarchies of folders under which real folders can be placed.

CompleteFTP administrators should carefully plan out their folder structure, and create virtual folders as appropriate. For example, a virtual folder called 'home' might be created at the root level, by selecting "Add root folder" and choosing the 'Virtual folder' option. Once created, its name may be modified from 'New Virtual Folder' by selecting this name and typing over it with the desired name.

A virtual folder is of little use unless a real folder is one of its children. Right- clicking on the folder will display a *New* menu that allows creating of subfolders beneath the current folder. For example, a Windows folder named after a user might be created here, so that it is under */home/username* in the hierarchy - even though it may exist at a location such as *D:\Users\username\ftphome* on the disk.

## How to add a Windows folder

Windows folders are standard folders on the local machine. They can be inserted into the virtual folder hierarchy at any point, including the root.

They are added by selecting "Add root folder" (or "Add sub-folder") and choosing "Windows folder". A folder browser dialog displays the available drives, and a folder can be chosen.

## How to add a Windows special folder

Windows special folders are folders that are predefined by Windows, and include folders such as "My Documents" and "My Music". They can be inserted into the virtual folder hierarchy at any point, including the root.

They are added by selecting "Add root folder" (or "Add sub-folder") and choosing "Windows special folder". A dialog box displays the types of special folders that are available.

It should be noted that only Windows users can access special folders. They have no meaning for non-Windows users, and an error is returned if a non-Windows user tries to change into a folder that is a Windows special folder.

# How to create a network/macro folder

Network/macro folders are basically folder names that are manually entered. They include network folders as well as "macro folders" - abstract folders that contain a macro for a username or domain name that is later expanded.

To add a new network/macro folder, select "Add root folder" or "Add sub-folder" and choose the 'Network/Macro folder' option. A dialog box will appear into which the Windows path of the folder can be entered (in standard format or in UNC format). For example, "\\machine1\mydir" could be the UNC path used.

Note that network folders in particular can introduce permissions problems. The user that is actually using a particular network folder must have Windows permissions to do so, otherwise the operation will fail. Similarly, the Windows permissions of the network folder may not be able to be inspected by the server if the user under which the server process runs (usually SYSTEM) cannot access the folder.

To enter a macro folder, use the arrow button on the right hand side of the dialog box to insert the macro of your choice. For example, "D:\users\%UserName%". Whenever this folder is accessed by a user the username is substituted in for the macro, so a "javaftp" user would be accessing "D:\users\javaftp".

# Gateway folders

A *gateway folder* is a directory on a different server to the current CompleteFTP server instance. It is basically a remotely mounted directory (analogous to a network share). This feature is only available in the Enterprise Edition of CompleteFTP.

Unlike normal network shares, gateway folders are specified by protocol - either FTP, FTPS or SFTP. This means that a remote folder from an SFTP or FTPS server can be accessed by logged in users as if it were a local folder on the CompleteFTP server.

So a user who logs into CompleteFTP via FTP is able to access the files on a different SFTP server if those folders are made accessible as a gateway folder. In effect, CompleteFTP does *protocol translation*. Some scenarios that are possible are listed here.

Gateway folders are created in the same way as other folders in the filesystem - via the *Folders* tab, and by choosing *New root folder* or *new sub-folder* - in the Enterprise Edition of CompleteFTP, there is a *Gateway folder* menu item. This brings up the *Configure Gateway Folder* dialog box.

Two options are available when configuring the new gateway folder:

1. Use the gateway connection. When a user logs into CompleteFTP via a gateway authenticator, they are authenticated by the remote server, and a connection to that remote server is stored in their session. This is the current gateway connection. By default, it uses the connection's working directory, although a specific directory on the remote server can be provided (as long as it exists and is accessible). This means the gateway folder is only accessible by users who are authenticated by the gateway authenticator.
2. Use a specified connection, i.e. specify the credentials to be used to access the gateway folder as part of the folder configuration. Of course, the protocol and remote server name must be also be specified. This means any user can access this gateway folder.

To use the gateway folder, users must change directory into it, or have the gateway folder set as their home directory. Gateway users require the *defaultExternal* user's home directory to be set to a gateway folder that uses the current connection.

# How to insert macros into folder-names

Macros may be inserted into the name of any folder in the virtual file-system. A common case where this may be useful is in setting the name of the home-folder of Automatic Windows Users such that the user-name matches the name of the folder that they see when they log in. To do this, change the name of the 'MyDocuments' folder to '%UserName%'.

To see the available macros, click the ellipsis (i.e. "...") button next to the name field in the folder properties panel. This shows a dialog containing a text-box with a small button to its right. Clicking this small button shows a list of the available macros.

## How to give every user the same directory structure

Users often require a group of users to share the same subfolders under their home directories, for example:

```
/JoeSmith
   /BusinessData
   /HumanResources
/FredJones
   /BusinessData
   /HumanResources
```

Here Joe Smith and Fred Jones are FTP users who need access to the same Windows folders, and want to see them as subfolders of their own home folders. JoeSmith is Joe's home folder, and FredJones is Fred's home folder. So here BusinessData and HumanResources, are required to be visibile as subfolders in Fred and Joe's home directories. They can, of course, be manually created for each user, but for many users this is very tedious.

One way to do this automatically is to set the home folder for each user to a single network/macro folder that has a macro in its path, and then adding the BusinessData and HumanResources folders beneath that folder. Here's how to do that:

1. Select the Folders panel.
2. Click 'Add folder', then 'Root folder' and then 'Network/macro folder'. In the dialog that pops up enter something like 'C:\FTPUsers\%UserName%' where 'C:\FTPUsers' is the directory under which you want your user's home directories created.
3. Check the 'Automatically create the directory'
4. Click OK
5. Select the folder you just created. It's probably called 'NetworkMacro Folder 1'. Click on the 'Name' property and change it to a meaningful name, such as 'UserHome'. Underneath '/UserHome' create folders that point to 'BusinessData' and 'HumanResources'.
6. Select the JoeSmith and FredJones users and set their 'Home Folder' to /UserHome

If you need your users to see a path other than /UserHome when they log in then you can set the name of that folder to '%UserName%'.

# How to set a folder's permissions

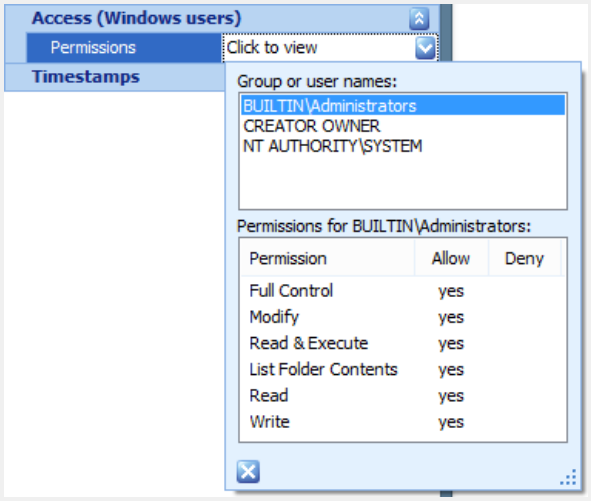
Access to files is controlled either by CompleteFTP itself or by the Windows file-system. For non-Windows users CompleteFTP will always control access. Windows users will be subject to Windows' own file-system permissions by default, though this can be changed (see below).

## User settings affecting file access

Access control for Windows users can be managed by CompleteFTP if the 'access control' property of those users has been manually set to 'non-Windows'. It's also possible restrict users to read-only access by setting the 'read-only' property of those users.

## Managing Windows permissions

While Windows file-system permission may be viewed from within CompleteFTP Manager (see 'Access (Windows users)') they can't modified from there. Windows Explorer or another tool must be used to edit Windows permissions




## Managing CompleteFTP (non-Windows) permissions

The *Standard Edition* of CompleteFTP features basic permission control, while the *Professional* and *Enterprise Editions* have more sophisticated permissions.

### Standard edition permissions

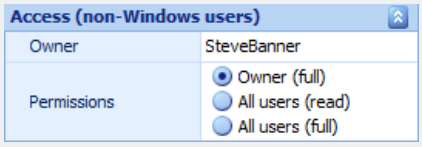
In the Standard edition, three levels of access are available for each folder:

**Owner (full)**
Owner has full permissions

**All users (read)**
Owner has full permissions, other users have read permissions

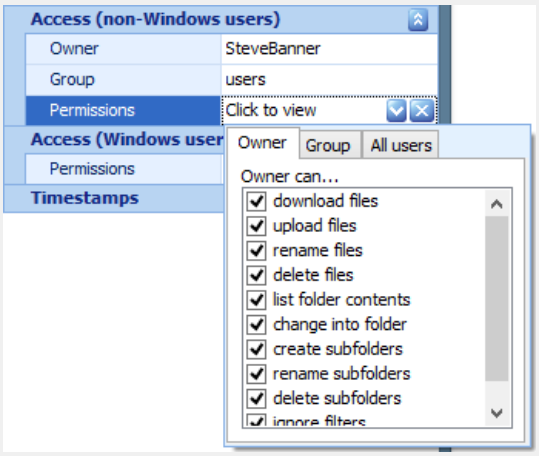
**All users (full)**
All users have full permissions

The owner of the folder can be selected in the Folder Properties section of the File System panel. The user permissions are shown as radio buttons under *Access (non-Windows users)*.




### Professional/Enterprise edition permissions

The Professional and Enterprise editions of CompleteFTP offer a permission system that's similar to that used in UNIX-like systems, such as Linux and OSX. This system was chosen because it's simple, flexible and widely known.




Permissions are managed in three different scopes: *owner*, *group* and *all users*. Each folder in the CompleteFTP's file-system will either inherit permissions from its parent folder or specify an owner, a group and a set of permissions for each of these scopes. Permissions are defined separately for the owner, the group and all users. The following permissions are available:

**download files**
: files may be downloaded/read from this folder

**upload files**
: files may be uploaded/written to this folder

**rename files**
: files may be renamed in this folder

**delete files**
: files may be deleted from this folder

**list folder contents**
: the contents of this folder may be listed

**change into folder**
: user may change into this folder

**create subfolders**
: folders may be created under this folder

**rename subfolders**
: subfolders of this folder may be renamed

**delete subfolders**
: subfolders of this folder may be deleted

**ignore filters**
: user is not subject to any file-name filters that are defined for this folder

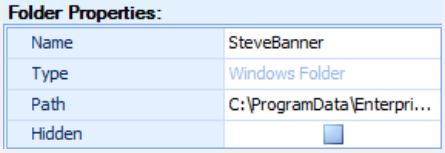
A user will have permission to perform a particular action if:

1. they are the owner of the group **and** that permission is enabled for the owner, **or**
2. they are a member of the group of that folder **and** that permission is enabled for the group, **or**
3. that permission is enabled for all users

The owner of the folder can be selected in the Folder Properties section of the Folders panel. The folder's group can be selected from the list of user groups. Any non-Windows user who is a member of that group will be granted the permissions that group has on the folder. Finally the permissions are shown. Click on "[Click to view]" and a popup will display the permissions for the folder's owner, group, and for all users. The permissions themselves are self-explanatory.

## Hidden folders

If the Hidden checkbox is checked then the folder will not show up in listings. This works independently of access permissions. For example, a folder may be shown but not accessible, or it may be hidden but still accessible.



# How to extend the filesystem (Enterprise Edition only)

The regular file-system can be completely replaced or extended by one of your own design. When a virtual folder is associated with your extension, your code will be called by the server whenever a client performs file operations (e.g. reading, writing, directory-listing) in that folder.

For example, your extension may read from and write to a database instead of the Windows file system.

See file system extensions for more details.

# How to stop executables being uploaded

Sometimes it is a requirement to block the upload of certain file types, such as executable files. This can be done by using filename filters, in the Filesystem settings of the site. Select the ellipsis to bring up the Filename Filters dialog, and add the filename masks required. For example, to match executable files, use "*.exe". Multiple filters can be added to match various file types.

The filters apply to all protocols - SFTP, SCP, FTP/FTPS and HTTP/HTTPS.

These filters can be applied in one of two ways. The default is to **block** the upload of filenames that match the filters. But this can be inverted by selecting the radio button to only **permit** files that match the filters.

It is important to realize that once set up, these filename filters apply to all files that are uploaded to the site - and to files that are being renamed (filename filters also prevent renaming files to banned names).

**Ignore Filters permission**

The only way to avoid filename filters is by specifically adding the "ignore filters" permission to a virtual directory. Then all users who have this permission will be able to upload any files they wish.

# How to mount ZIP files as directories

ZIP files contain many files combined into one compressed archive. When residing on a file server, normally the entire ZIP file would need to be downloaded to access a single file within the ZIP file.

CompleteFTP 7.1 introduces ZIP file navigation. When ZIP file navigation is enabled ZIP files are presented to clients as though they are folders. This means that users can change into folders within the ZIP file and download specific files. This can save time and bandwidth as it allows users to download only what they need from a ZIP file instead of downloading the whole ZIP file.

By default, both the ZIP file itself and the folder that corresponds to the ZIP file are shown in listings, the latter having the additional extension '.folder'. The additional extension can be changed by modifying the 'ZIP-folder extension' setting. If this setting is cleared completely then the ZIP file won't be displayed being replaced by the corresponding folder.

Currently ZIP files cannot be updated on the server - they are read only. This feature may be added to a future release.

See the filesystem settings for more details.

# How to enable backslash separators

Most FTP servers including CompleteFTP use Unix-style forward slash separators in paths. Occasionally it is desirable to be able to use Windows-style backslash separators - for example if there are clients accessing the server that only support backslash separators. CompleteFTP does not support backslash separators by default, but enabling this option enables support for backslash separators (forward slash separators are still supported if this option is enabled).

See the filesystem settings for more details.

# Users

CompleteFTP supports unlimited numbers of users, including Windows and non-Windows users.

The *Users* screen displays the current users, and ways to add, modify and delete users.

If an existing user's details are selected in the list of users shown, a panel at the right enables the details to be changed, such as the username, password, and home directory. The allowed protocols for the user are also displayed - as well as on a site basis, protocols can be enabled or disabled for each user. The user can also be disabled by unchecking the *Enabled* checkbox.

Note that CompleteFTP supports the Automatic Windows Users (AWU) feature. This allows the administrator to enable login of any Windows user without having to manually add the user to CompleteFTP's internal user list. See how to configure automatic Windows users for more details.

# General user settings

The general user settings that are available to configure *CompleteFTP* sites are described below. They are found on the *Users* tab, and can be displayed by selecting the *General user settings* link. Note that settings must be saved via the *Apply Changes* button, otherwise they will be lost.

**General settings**

| Setting | Description |
|---|---|
| Allow anonymous FTP users | If checked, anonymous access is enabled. This means connections can be made by logging in as an anonymous user rather than as a named user with a password. Anonymous users are represented by the user called 'anonymous'. By default the home-folder of this user is /Public, so this is the folder that will be publicly accessible if this option is enabled. |
| Allow public HTTP access | If checked, unauthenticated users will be able to access any file in the public folder. Unauthenticated users are represented by the user called 'anonymous'. By default the home-folder of this user is /Public, so this is the folder that will be publicly accessible if this option is enabled. |
| User's home folder appears as root | If this option is enabled then the user will see their home directory as the root (i.e. /). If it is disabled then the user will see the absolute virtual file-system path, e.g. "/home/javaftp". By default this option is not checked. |
| Default Windows domain | If a default Windows domain is set, it means that Windows users on the default domain can login without specifying the domain (instead of using MyDomain\MyUserName they can use MyUserName). This can be very convenient for users. The default Windows domain tells CompleteFTP which domain to authenticate against if (1) a domain-name was not given and (2) no exact match was found in CompleteFTP's list of users. See How to set a default Windows domain. |

**Other authentication methods**

Users may be authenticated by other methods such as:

- **Automatic Windows** (all editions) - enables login of any Windows user without having to manually add each user to CompleteFTP's internal user-list (see here).
- **Database** (Professional and Enterprise editions) - authenticates against an external data-source such as a database table or spreadsheet (see here).
- **Gateway** (Enterprise edition) - authenticates via another FTP or SFTP server (see here).
- **Custom** (Enterprise edition) - write your own authentication method using any .NET language (see here).

Users that are authenticated via one of these methods are subject to the settings of the specified 'log-in-as' user. For example, if Automatic Windows users are only to be allowed to connect via SFTP then only SFTP should be enabled for the log-in-as user that's defined for the Automatic Windows authenticator, which is the defaultWindows user by default.

**Password policies**

Password policies can be found by selecting the "Password policies" link on the General User Settings dialog box, and are described here.

# How to add a user

Both Windows and non-Windows users can be added to CompleteFTP.

By default, they will all be assigned home directories within the special folder for Common Application Data. On XP this is often *C:\Documents and Settings\All Users\Enterprise Distributed Technologies\Complete FTP\Users*, while on Vista it is something like C:*ProgramData\Enterprise Distributed Technologies\Complete FTP\Users*.

See how to add a Windows user for Windows users. Windows users can also be enabled automatically - see how to configure automatic Windows users.

See how to add a non-Windows user and how to add multiple users for non-Windows users, and How to configure external users for users stored in an external database.

# How to add a Windows user

To add an existing Windows user, click the *Add user* link in the Users panel and then select *Windows user(s)* from the menu. When adding Windows users from the local server then click *List server's local users*. When adding Windows domain users (not available in Standard Edition) enter the domain name and click *List domain users*. For domains with a very large number of users the size of the list can be restricted and/or a filter can be provided to reduce the number of users listed.

Once the list of users has been fetched and displayed, select the users to be added from the list. Hold down the shift or control keys to select multiple users.

If a single user is being added and the checkbox labeled *Automatically create home folders* is checked then a home-folder with the same name as the user will be created automatically. The folder will be created under the `/Home` folder in the virtual file-system, and by default will be mapped to a Windows directory underneath the directory to which `/Home` is mapped. The mapping can be changed by clicking the ellipsis (...) to the right of the label of the checkbox. By default, the mapping is created using the template `%HomeBaseFolder%/%UserName%`, but this can be changed as desired.

If multiple users are selected when the *Add selected user(s)* button is clicked then a dialog-box will pop up. In this dialog-box you can choose to create a home-folder in the same manner as described above for single users, or you can choose to use the same directory for all users.

Once all users have been added click *Close* to close the dialog.

Back in the Users window, all the users that have just been added will be selected, so that any properties that must be set commonly for all users may be set.

# How to add a non-Windows user

*Add a non-Windows user* should be selected to add a user that does not exist as a current Windows user on the server machine (or domain). This means an FTP user does not have to exist as a Windows user. The username, password and details should be entered in the appropriate dialog boxes.

In step 3, the home directory for this user must be selected. By default, the new user's home directory will be created in the Home directory (see how to add a user for more details).

If a different home directory is desired, use 'Select non-default home directory' to choose one of the folders listed. You may need to create a virtual folder for them.

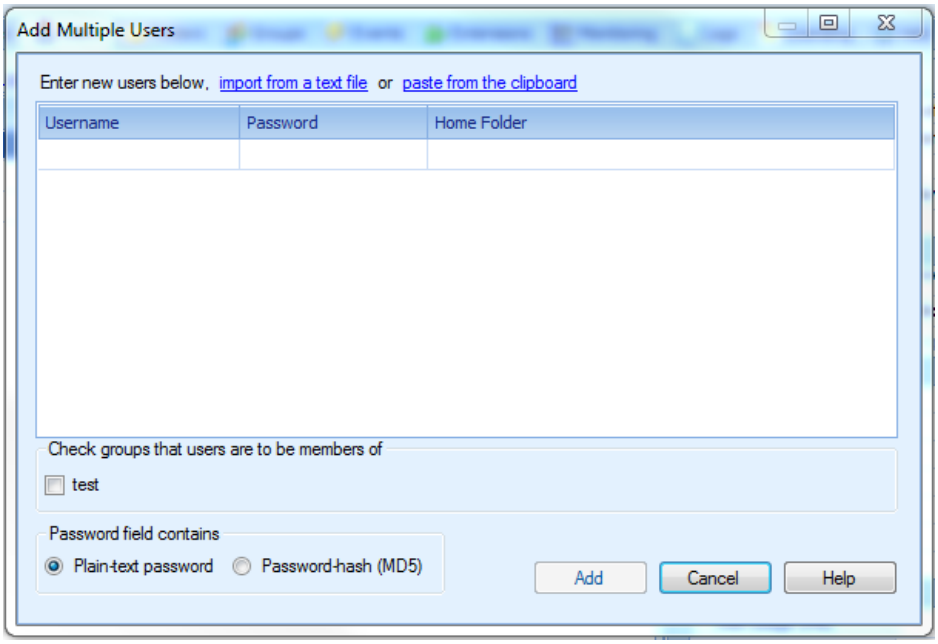
After selecting the home directory, select the 'Next' button, and finally 'Apply' to add this new user to the server database.

If the user's home directory already exists, a dialog will warn of this. Select 'Yes' to reuse the folder.

Back in the Users window, which displays the current users, note that selecting on any user displays their details in the right hand panel. Any of these details can be edited at any time (although any changes will not apply to currently logged in users).

# How to add multiple users

Often, it is desirable to add multiple users at the same time rather than one by one. This can be done for non-Windows users by selecting *Add user* and choosing the *Multiple non-Windows users* menu item. This displays a dialog that allows multiple non-Windows users to be inputted by various means (displayed below).




Users can be explicitly typed in, imported from a text file, or pasted from the clipboard. For the latter two cases, the text file or clipboard data needs to have a user per line, and fields must be separated by commas or tabs. The first field contains the username, the second field the password, and the third field the Windows path of the home folder. Only the username is mandatory. If the password is supplied as an MD5 hash rather than plaintext, then the MD5 radio button should be selected.

Finally, if the administrator has created any groups, they are displayed and can be checked so that all added users are members of the checked gruops.

# How to change user properties

User properties are modified by selecting the user's row in the list of users displayed in the *Users* tab. The user properties are displayed on the right. After properties have been changed *Apply Changes* must be selected to commit the changes to the server.

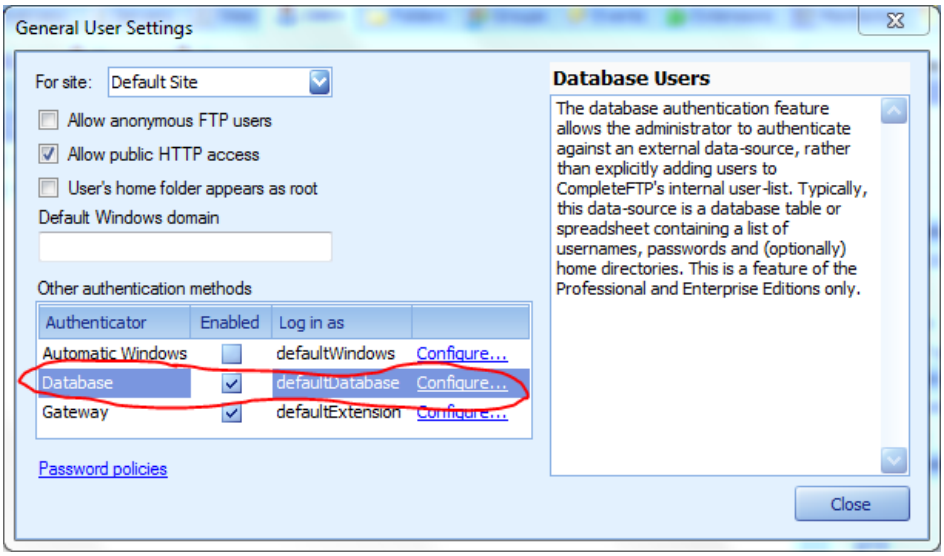
**Multiple users** can be selected from the list of users by using the combination of the control key and left button click. The properties panel displayed can be used to set properties for all of the selected users.

Multiple users can also be selected via the right mouse button menu item, *Select*. All users can be selected, or users from a particular group or matching a wildcard. Additional groups can be created from this menu as well, using the *Groups* menu item.

# How to configure external database users

The database authentication feature allows the administrator to authenticate against an external data-source, rather than explicitly adding users to CompleteFTP's internal user-list. Typically, this data-source is a database table or spreadsheet containing a list of usernames, passwords, optionally home directories and groups. This is a feature of the Professional and Enterprise Editions only.

Database authentication may be set up in the General User Settings dialog, accessed from the User panel. The Database authenticator must be enabled and configured via the 'Configure' link shown below


General User Settings
For site: Default Site
Allow anonymous FTP users
Allow public HTTP access
User's home folder appears as root
Default Windows domain
Other authentication methods
Authenticator
Enabled
Log in as
Automatic Windows
defaultWindows
Configure...
Database
defaultDatabase
Configure...
Gateway
defaultExtension
Configure...
Password policies
Database Users
The database authentication feature allows the administrator to authenticate against an external data-source, rather than explicitly adding users to CompleteFTP's internal user-list. Typically, this data-source is a database table or spreadsheet containing a list of usernames, passwords and (optionally) home directories. This is a feature of the Professional and Enterprise Editions only.
Close

In the database authentication configuration dialog, enter the OLE DB connection-string in the top-most text-box and press the 'Test connection' button to verify that it works. The website, connectionstrings.com, is an excellent resource for connection-strings. It lists various types, so remember to choose an OLEDB connection-string. Also remember that the connection is made from the server, so if you are using CompleteFTP Manager on a machine other than the server then the connection-string may differ from the one you would use on your local machine.

The SQL query retrieves the user's password/password-hash. It may also optionally define a Windows home-folder and a list of groups to which the user belongs.

The password or password-hash must be the first field returned. It is highly recommended that any passwords stored in a database are hashed. Even better, use a 'salt' prepended to passwords before they are hashed. See the fourth field, below.

If a second field is returned then it will be assumed to be the Windows path of the user's home-folder. If the home-folder doesn't already exist when the user logs in then it will be created at that time. Note that this field will not be used if the type of the home-folder of the log-in-as user is "Virtual folder".

If a third field is returned then it will be assumed to be a comma-separated list of group-names (e.g. MyGroup1,MyGroup2,MyGroup3). The user will be treated as a member of all of the groups listed as well those of which the "defaultDatabase" user is a member.

If a fourth field is returned then it will be assumed to be the 'salt' that was prepended to the password before it was hashed (if hashing is enabled). The salt should be a random string of at least 8 characters. Salting is intended to defeat rainbow table attacks. Rainbow tables are sets of pre-calculated hashes for common passwords. If salts are not used, attackers can simply match these hashes with the hashes in your database. If salts are used, a new rainbow table must be calculated for each salt value (and these tables are computationally expensive).

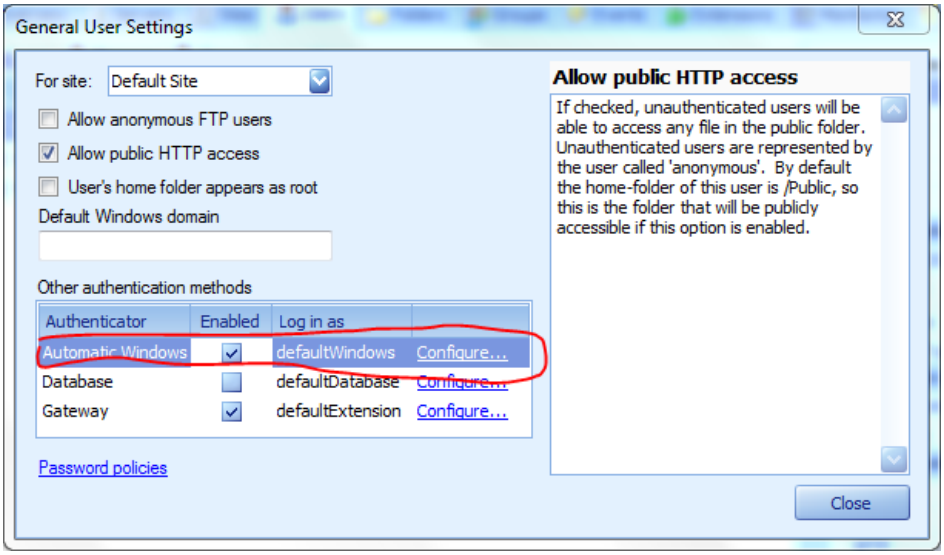
Users connecting via this method are subject to the settings of the "defaultDatabase" user, which may be found in the Users panel. The administrator may, for example, enable only certain protocols for Automatic Windows user connections. If distinct settings are required for specific users then those users should be added explicitly as Windows users in the Users panel.

By default, the home folder is shown as /DatabaseUser. If the home folder is not provided by the query, the value of this folder will be used (which is a macro folder by default). If a home folder is provided by the query, the query's value will be used, but /DatabaseUser will still be displayed as the name of the user's home directory. The displayed name can be changed by modifying the name of the DatabaseUser system folder to use a macro, as described here.

# How to configure automatic Windows users

The Automatic Windows authentication method allows the administrator to enable login of any Windows user without having to manually add each user to CompleteFTP's internal user-list. It is configured in the General User Settings dialog, accessed from the User panel.

The feature may be enabled by checking the checkbox in the Enabled column. In the Professional Edition (or better) logins may be restricted to one or more Active Directory (AD) or local groups by selecting the Configure ... link.


General User Settings
For site: Default Site
Allow anonymous FTP users
Allow public HTTP access
User's home folder appears as root
Default Windows domain
Other authentication methods
Authenticator
Enabled
Log in as
Automatic Windows
defaultWindows
Configure...
Database
defaultDatabase
Configure...
Gateway
defaultExtension
Configure...
Password policies
Allow public HTTP access
If checked, unauthenticated users will be able to access any file in the public folder. Unauthenticated users are represented by the user called 'anonymous'. By default the home-folder of this user is /Public, so this is the folder that will be publicly accessible if this option is enabled.
Close

Users connecting via this method are subject to the settings of the "defaultWindows" user, which may be found in the Users panel (and by selecting "Show system users"). The administrator may, for example, enable only certain protocols for Automatic Windows user connections. If distinct settings are required for specific users then those users should be added explicitly as Windows users in the Users panel.

By default, the home folder is `/MyDocuments`, which resolves to the logged-in user's Windows My Documents folder. This may be changed by modifying the Home Folder property of the `defaultWindows` user. If a separate folder (other than `/MyDocuments`) is required for each folder then a Network/Macro Folder should be used. This type of folder allows the insertion of a `%UserName%` macro into a path, which is replaced by the logged-in user's user-name. To do this, create a new virtual folder of the type Network/Macro Folder and set the Home Folder property of the `defaultWindows` user to this folder. The Network/Macro Folder's path could, for example, be set to `H:\Homes\%UserName%`, which would mean that when a user called Joe logs in their home folder would effectively be `H:\Homes\Joe`.

# How to configure gateway users

Gateway users are an Enterprise Edition feature of CompleteFTP. Gateway users are users who are authenticated by a different server - often residing in a different location.

Gateway users are analogous to automatic Windows users - they permit the login of any user located on a remote machine (rather than the current machine) without having to manually add each user to CompleteFTP. The remote users must be accessible via FTP, FTPS or SFTP on the remote machine.

To authenticate users on a different server, a gateway authenticator must be configured as described here.

Users connecting via this method are subject to the settings of the "defaultExternal" user, which may be found in the Users panel (and by selecting "Show system users"). The administrator may, for example, enable only certain protocols for gateway user connections.

If gateway users are to access the remote filesystem that they are authenticated for, they need to change directory into a gateway folder or have their home directory set to the gateway folder corresponding to that server. In the latter case, the *defaultExternal* user must have its home directory set to the gateway folder. If the gateway folder set as the home directory is configured to use the gateway connection then each user authenticated on the remote server will get their own home directory on the remote directory (the same directory they would have if they logged into the remote server directory instead of via CompleteFTP).

# How to set user quotas

CompleteFTP (Professional and Enterprise Editions only) supports the use of disk quotas and bandwidth quotas, so that users can be restricted in their use of system resources.

By default, no quotas are enforced - they have to be set explicitly for each user. This is done by going to the *Users* panel in the CompleteFTP manager, and setting the values in the *Quotas and Limits* section amongst the properties that are listed.

Disk usage is set in megabytes (MB) and is a cumulative limit across all sites.

Upload and download bandwidth limits are set in kilobytes/second (kB/s). Note that bandwidth is **not** set in kilobits/second (kbs), and that bandwidth limits apply per site (so that in the Enterprise Edition, if a user is enabled on multiple sites then their total bandwidth is [number of sites] x [set bandwidth]).

# How to set up public key authentication

Users connecting to CompleteFTP via SFTP or SCP (supported in the Professional and Enterprise Editions) can authenticate themselves in two different ways (if the server permits it).

- *Password authentication* simply uses their existing username and password
- *Public key authentication* requires the client to register a public key with the server, and to supply the corresponding private key when logging in. The private key is not sent to the server, but is used to sign certain data that the server confirms (using the registered public key) was signed by the client.

## Enabling public key authentication

There are two places public key authentication is enabled. The Site's authentication methods is the primary place, in the SFTP/SCP/SSH settings. Public key authentication is enabled by default.

Public key authentication uses either the RSA or DSA algorithm, and this can also be set via SFTP->Advanced SFTP Settings->Algorithms. Normally the default of All is used.

The other place is in each user's properties (Enterprise Edition only). In the Authentication section, the SSH methods can be specified. They are combined with the Site authentication methods. A method has to be set for both the user and site to be available for that user.

## Setting up the user's keys

For a user to use public key authentication, a keypair (consisting of a private key and a public key) must be generated, and the public key registered with CompleteFTP. Both RSA and DSA can be used.

Select the user in the Users window, and then select the elipsis (...) on the right of "Public keys" in the user details window. This brings up the "Manage Public Keys" dialog, which displays the currently loaded public keys for this user.

Use "Generate a new keypair for this user and add it to the list" to create a new keypair. Make sure the *private key* you are prompted to save is kept securely - it is required by clients to authenticate as this user.

You can also use "Import a public key and add it to the list" to import existing RSA or DSA public keys for this user. Public keys can also be deleted and exported. Note that the Professional and Enterprise Editions permit multiple RSA and multiple DSA keys for each user.

# How to control user access by IP address

It's possible to configure CompleteFTP to allow specific users to connect only from specific IP addresses. This is done via the user property called 'Allowed IP addresses' in the User Properties panel, but for this to have any meaning it's necessary to firstly deny access from other IP addresses.

**Denying access from other IP addresses**

Denying access from IP addresses other than those allowed for specific users must be done via the IP filtering property in the Site Properties panel, as follows:

1. Open the Site Properties panel
2. Expand the IP Filtering and Auto-banning category
3. Select the IP filtering property
4. Click the ellipsis button on the right
5. Select 'Allow over deny'
6. If there's currently a rule that allows all connetions, change this to deny all.

Once this is done, no connections will be allowed from any IP address. Note that more information on IP filters may be found in the section on IP Filtering and Auto-Banning.

**Allowing specific IP addresses for specific users**

Now that CompleteFTP has been configured to deny connections by default, we can go ahead and allow specific users to connect from specific IP addresses, as follows:

1. Go to the Users panel
2. Select one or more users
3. If you're in the Overview panel, right-click and select Properties to open the User Properties panel.
4. Select the property labelled 'Allowed IP addresses'.
5. Enter the IP address(es) from which the user is to be allowed to connect. Separate multiple IP addresses with commas. Domain-names may also be used.
6. Apply the changes

Note that entering allowed IP addresses simply automatically creates the appropriate user-specific IP filter rules (see here). More specifically, an 'allow rule' is created for the specified IP addresses the users on each site on which the user is enabled. Further customization of these rules may be done via the Site Properties panel.

# How to set up client certificates (FTPS)

For FTPS, clients can be forced to supply a valid client certificate (not available in the Standard Edition). This setting can be found by navigating to Settings->FTP/FTPS->Advanced FTP/FTPS Settings->Security Settings. The "Clients must supply certificates" setting is set to "Don't require valid certificate" by default, but can be changed to "Require valid certificate".

If a valid certificate is required, all FTPS clients must send a certificate that may be validated against the Windows certificate store. This means that either the client's certificate itself, or the certificate of one of its issuers must be in the Windows 'Trusted root certificate authorities' store. This store may be managed using the Windows Administrative Tools, or by clicking '...' to the right of this security setting.

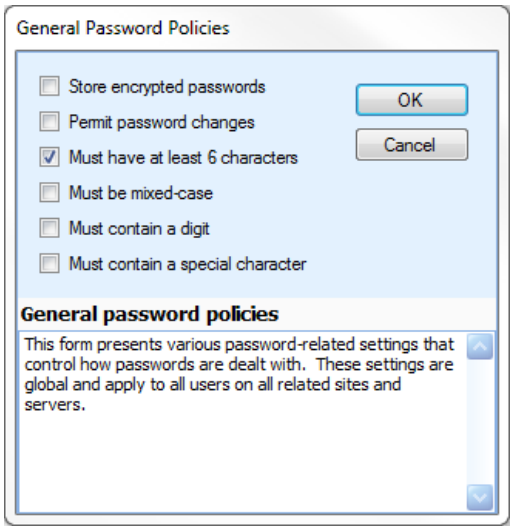
## How to make a user's home dir root

Typically users see their home directory as a path, if the defaults are used. This is the absolute virtual filesystem path, e.g. "/home/javaftp".

It is sometimes desirable to make the user's home directory the root, i.e. "/". This is a configuration setting for the site. See the *Settings* panel's General User Settings for more details on enabling this feature.

# How to set password policies

System-wide password policies are accessible from the *Password policies* link on the General User Settings dialog box, found in the *Users* panel. The Password Policies dialog is shown below. For security, it is recommended that *must have at least 6 characters*, *must be mixed-case* and *must contain a digit* are enabled.


General Password Policies
Store encrypted passwords
Permit password changes
Must have at least 6 characters
Must be mixed-case
Must contain a digit
Must contain a special character
OK
Cancel
General password policies
This form presents various password-related settings that control how passwords are dealt with. These settings are global and apply to all users on all related sites and servers.

Store encrypted passwords is not enabled by default. Usually, passwords for non-Windows users are not stored for security reasons. Instead, a hash of the password is stored, and when users log in, the hash of the password they supply is compared with the stored hash.

If *Store encrypted passwords* is enabled, passwords for non-Windows users will be stored in encrypted form. This is less secure than using a hash, but does mean that the adminstrator is able to retrieve passwords.

Note that Windows users do not have passwords stored by CompleteFTP in any way. Instead, standard Windows authentication is used.

Permit password changes is not enabled by default. If it is enabled, non-Windows users can use the SITE command CPWD to change their password. SFTP also has a mechanism permitting the changing of a password (which must be supported by the SFTP client).

Must have at least 6 characters is enabled by default. Must be mixed-case, must contain a digit, and must contain a special character are not enabled by default.

# File sharing

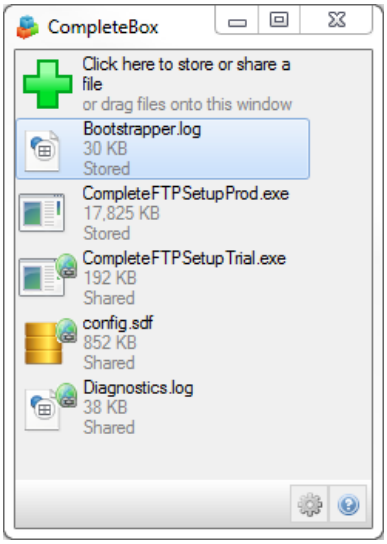
*CompleteBox* is a application installed on user's computers that allows them to easily store and share their files on a server machine running CompleteFTP.

This is known as *collaborative file sharing and storing*, and it requires CompleteFTP in either the Professional or Enterprise Editions, running at least version 8.0.

To store or share files, file sharing must first be enabled in CompleteFTP. Users install *CompleteBox* and use it to share their local files. Files can be stored on the server for private use, or shared with others.

By default, both the Professional Edition and the Enterprise Edition include one user license for sharing. If further users are required, they can be purchased here.

The CompleteBox desktop application is shown below:



# How to enable file sharing

*CompleteBox* enables *storing or sharing* of local files on a CompleteFTP server instance, which must be either the Professional Edition or the Enterprise Edition.

There are a few settings that must be enabled in CompleteFTP for CompleteBox users to begin using CompleteBox to share or store their files. If file-sharing is enabled during installation, all the settings below should already be enabled. The settings below should be checked if there are any problems getting file-sharing to work.

1. File sharing must be enabled in CompleteFTP. The CompleteFTP site must have sharing enabled, as described here. If it's turned off then users will no longer be able to connect via CompleteBox and shared files will no longer be available via HTTP/S.
2. Public HTTP access must be enabled, as described here. It is enabled by default. If it is disabled then files that have been shared via a link will not be accessible to users who have not logged in.
3. HTTPS must be enabled, as described here. It is enabled by default. If it's disabled then shared files will not be accessible via a link.
4. The anonymous system user must be enabled, via the Users tab. Use 'Show system users' to display it. It is enabled by default. As with public HTTP access above, if it is disabled then files that have been shared via a link will not be accessible to users who have not logged in.
5. If a user is to share files, then their account must be enabled for sharing and they must be a member of the *users* group. If a user's sharing is disabled then they will no longer be able to connect via CompleteBox and any files that they have shared will no longer be accessible.

Finally, sufficient CompleteBox user licenses must be installed in CompleteFTP so that every user who is file sharing has a user license. Each person requires a user license. By default, a one user license is included. If further users are required, they can be purchased here.

# How to install CompleteBox

To share a local file, the *CompleteBox* client must be installed on the local machine that files are being shared from.

**Getting the installer**

The CompleteBox installer is shipped with CompleteFTP server. After CompleteFTP has been installed, the CompleteBox installer can be found from the CompleteFTP program group in the Windows Start Menu - there is a link to the CompleteBox Installer folder, and *CompleteBoxSetup.exe* can be found there.

The installer can also be downloaded from your company's CompleteFTP website. On the default index page of the server, a link "Get started with file sharing" provides installer links.

The CompleteBox installer can also be downloaded directly from the EnterpriseDT website, here.

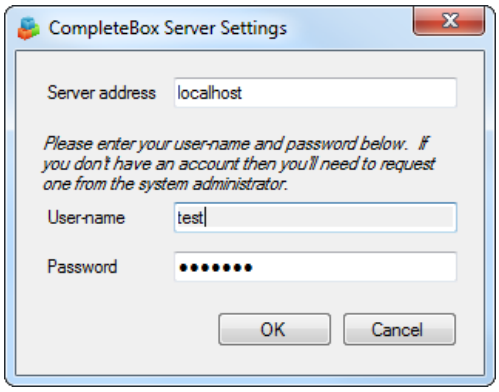
The installer can freely redistributed, although it may only be used with a CompleteFTP server.

**Running the installer**

The installer is a executable that requires administrator permissions to run.

If you are upgrading, it is important to upgrade CompleteFTP **first**. Older versions of CompleteBox will work with the latest version of CompleteFTP (as well as previous versions of course), but the latest version of CompleteBox **requires** the latest version of CompleteFTP.

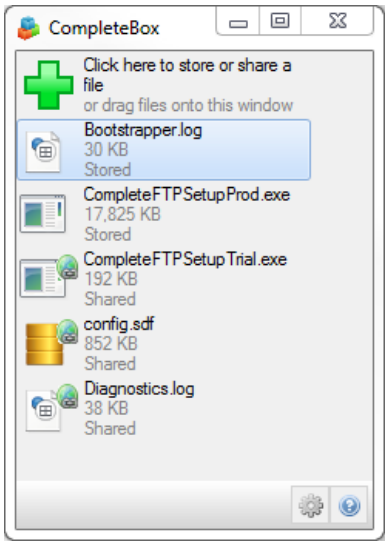
Once installed, *CompleteBox* runs as a Windows system tray application. It can be opened by clicking on its icon in the Windows system tray.




An account on the file sharing server is required, and the credentials must be entered when prompted. The file sharing server must have sufficient user licenses free to permit an additional file sharer. The server settings dialog is shown below:

Once installed, files can now be *stored or shared*.

## File sharing vs file storing




*CompleteBox* is an application installed on user's computers that allows them to easily store and share their files on a CompleteFTP server machine set up for this purpose.

*Sharing* a file means that the file is uploaded to the server and a public URL is generated by which others can download the shared file. You share a file so others can have access to it. A shared file is marked as "Shared", and has a link icon superimposed on its file icon.

*Storing* a file means that the file is uploaded to the server for the user's private use. No URL is available, and no-one other than the user can access the file. A stored file is marked as "Stored".

**Why store and not share?**

There are two reasons why a CompleteBox user might use file storage without sharing.

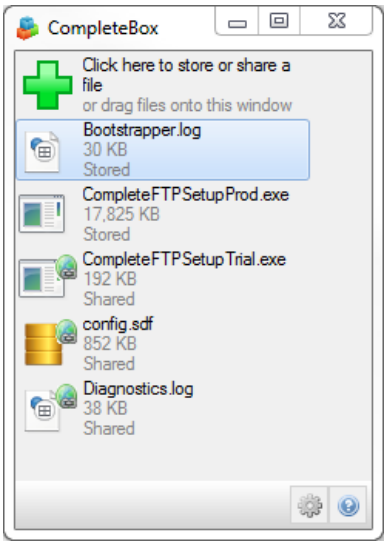
**Backup**. The first reason is that it is a useful way to quickly back up files onto a server. Select and upload, and your local files are now safely stored in case of a local machine failure.

**File synchronization**. The second reason is for private synchronization of files between machines. If you work in two locations (e.g. a work location and a home location), you can upload from one machine, and later download onto another. CompleteBox file synchronization will help you to keep files on both machines identical.

# How to share files with CompleteBox

Sharing a file with *CompleteBox* means that the selected file is uploaded to the server and a public URL is generated by which others can download the shared file. Provided someone is given the URL, the file is publicly available.

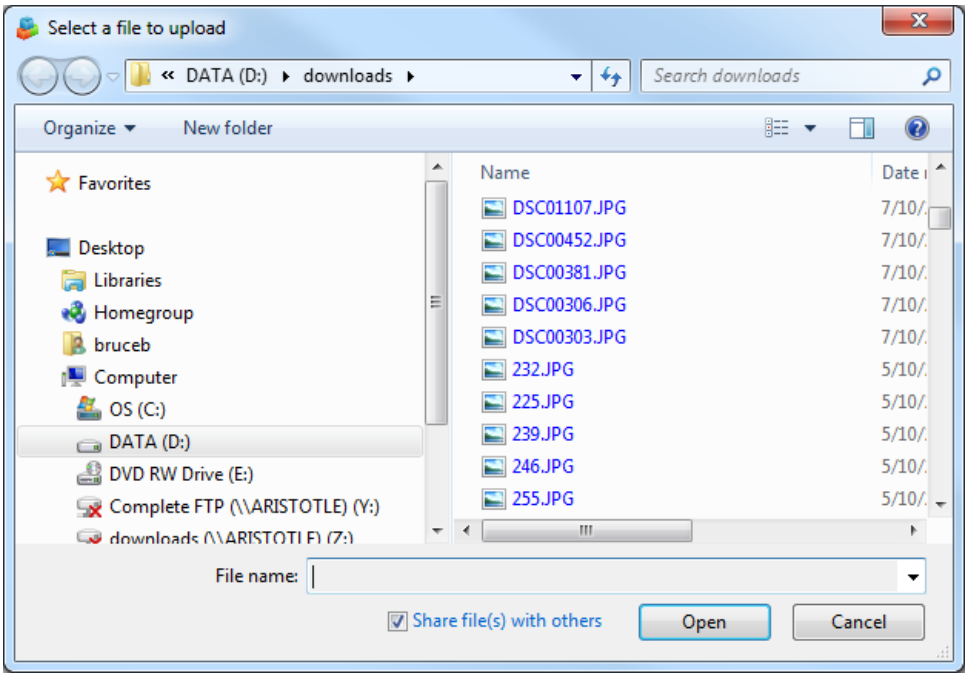
To share a local file, the *CompleteBox* client must be installed on the local machine that files are being shared from, as described here. The main dialog is displayed by selecting the CompleteBox icon in the Windows system tray, shown below.




Click on the green cross to display a file open dialog box for selecting files to share, or drag and drop files onto the dialog. Make sure the "Share file(s) with others" checkbox at the base of the dialog is ticked - if it is not, the files are stored but not shared. Sharing is the default.

Multiple files can be selected in the file open dialog by holding down the control key and selecting each file. Once selected, the files will be shared on the file sharing server via a secure connection. A unique link (i.e. a URL) is generated that can be emailed to the end-user of the file. Links are configured to expire in 30 days by default. If a single file is shared, the share URL is sent to the clipboard.

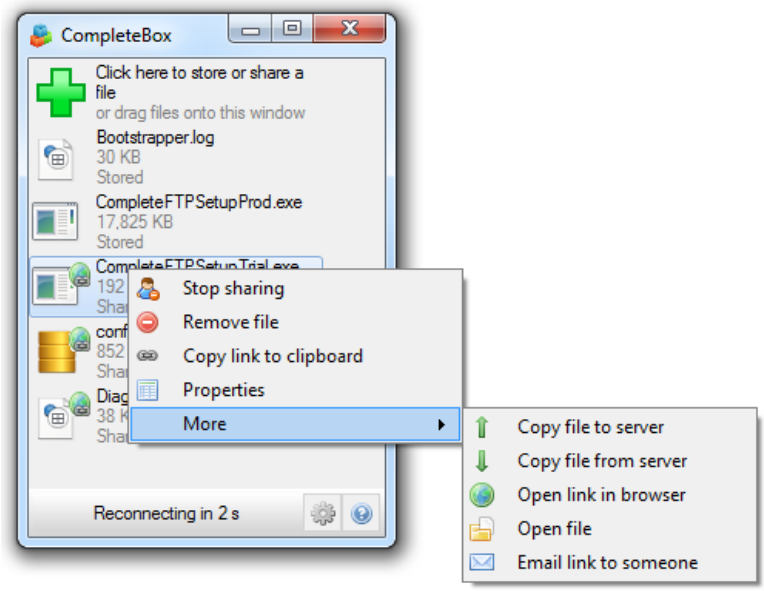
Note also that files can be drag-and-dropped or copy-and-pasted onto the main CompleteBox window, and Windows Explorer has a menu item "Store in CompleteBox" which will store or share the selected files.




Right clicking on a shared file in the list will display the right-click menu shown below, which allows the sharing to be stopped, the file removed, the link copied to the clipboard, and the file properties to be displayed. Multiple files can be selected by holding down the control key and selecting each file. Control-A will select all files listed.

Additional options allow the forced copying of the local file to the server, or from the server to local file. This is in case you know the local or remote file to be the authorative one, and you want the other copied over, irrespective of which is the most recent file. You can also open the link in a browser, open the local file, or email the URL to someone.

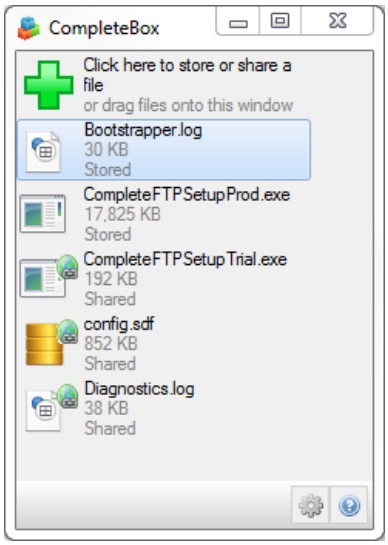
Finally, Control-C can be used to copy the selected files to the clipboard. Select a target directory in Windows Explorer, and Control-V will paste copies of these files into the directory.


CompleteBox
Click here to store or share a file
or drag files onto this window
Bootstrapper.log
30 KB
Stored
CompleteFTPSetupProd.exe
17,825 KB
Stored
Stop sharing
Remove file
Copy link to clipboard
Properties
More
Copy file to server
Copy file from server
Open link in browser
Open file
Email link to someone
Reconnecting in 2 s

# How to store files with CompleteBox

Storing a file with *CompleteBox* means that the selected file is uploaded to the server, but no URL is generated and the file is not available to anyone except its owner. Typically this is done for backup, or for a user to share their own files with themselves on two different computers.

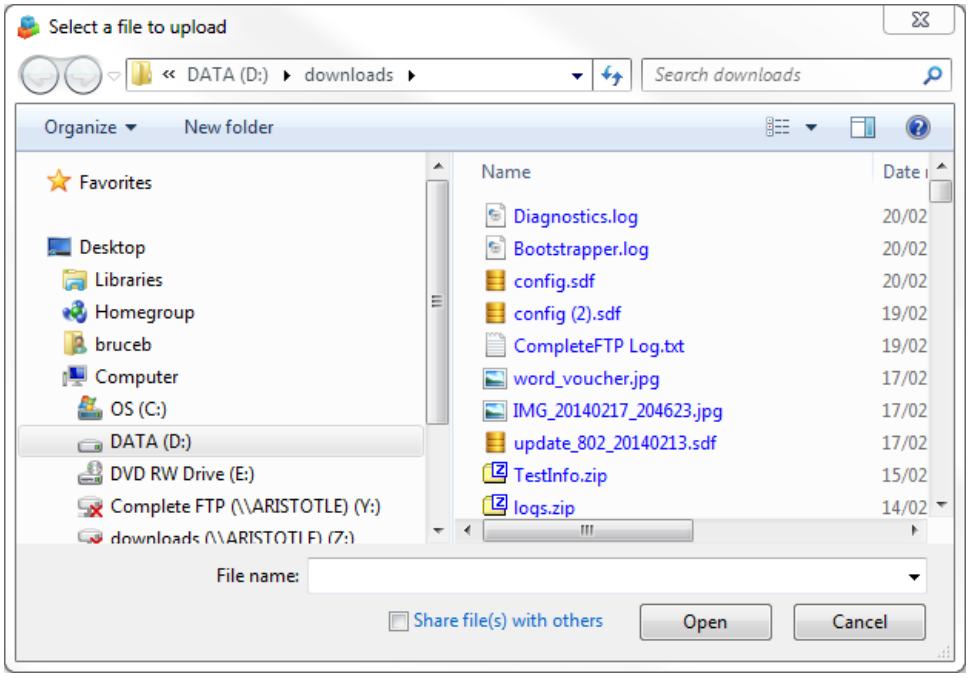
To store a local file, the *CompleteBox* client must be installed on the local machine that files are being shared from, as described here. The main dialog is displayed by selecting the CompleteBox icon in the Windows system tray, shown below.




Click on the green cross to display a file open dialog box for selecting files to share, or drag and drop files onto the dialog. Make sure the "Share file(s) with others" checkbox at the base of the dialog is **unticked** as below (otherwise the file will be shared).

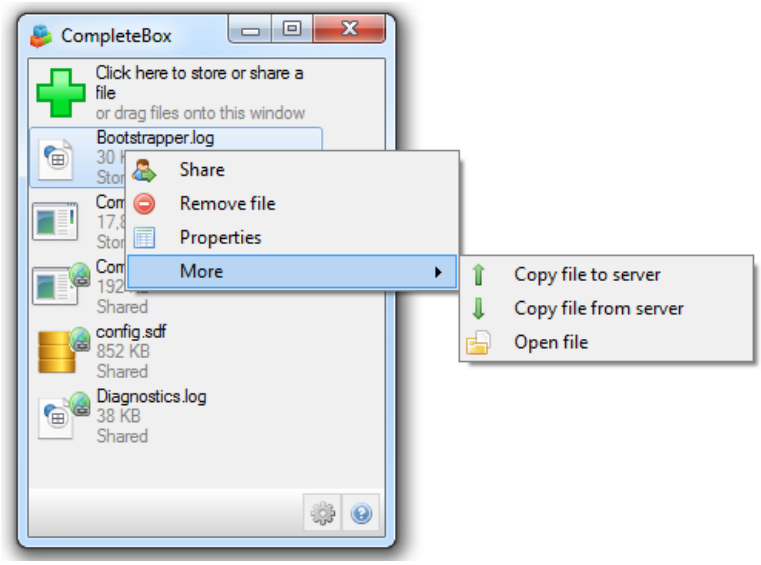
Multiple files can be selected in the file open dialog by holding down the control key and selecting each file. Once selected, the files will be stored on the file sharing server via a secure connection.

Note also that files can be drag and dropped onto the main CompleteBox window, and Windows Explorer has a menu item "Store in CompleteBox" which will store or share the selected files.



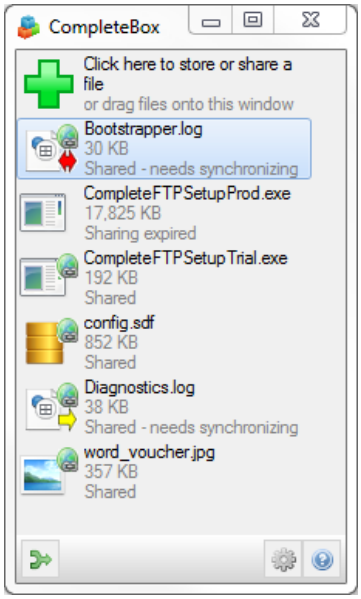

Right clicking on a stored file in the list will display the right-click menu shown below, which allows the stored file to be shared, the file to be removed, and the file properties to be displayed. Multiple files can be selected by holding down the control key and selecting each file. Control-A will select all files listed.

Additional options allow the forced copying of the local file to the server, or from the server to local file. This is in case you know the local or remote file to be the authorative one, and you want the other copied over, irrespective of which is the most recent file. You can also open the local file.


CompleteBox
Click here to store or share a file
or drag files onto this window
Bootstrapper.log
Share
Remove file
Properties
More
Copy file to server
Copy file from server
Open file
Shared
config.sdf
852 KB
Shared
Diagnostics.log
38 KB
Shared

# File synchronization

Once a local file has been shared or stored on the server, there is the possibility that either copy will become out of date. This means the file needs to be *synchronized*, i.e. the local copy and the server copy need to be made identical.




This can happen in two ways. The local file could be modified by the user, or the remote file could be modified. The remote file could be modified if the same user uses CompleteBox on a different machine to upload a change. If this happens, CompleteBox will show that the files are different by displaying a "needs synchronizing" message under the filename, and changing the icon for the file.

A red double arrow means both the local and remote file have changed since the last synchronization. In this case, you will be prompted to select which file should be the authorative one.

A yellow arrow means either the local or the remote file have changed since the last synchronization. When you synchronize, the most recently changed file will be copied over the older file. The arrow indicates the direction of the copy. Pointing to the right means the local file will be copied over the server file, and pointing to the left means the server file will be copied over the local file.

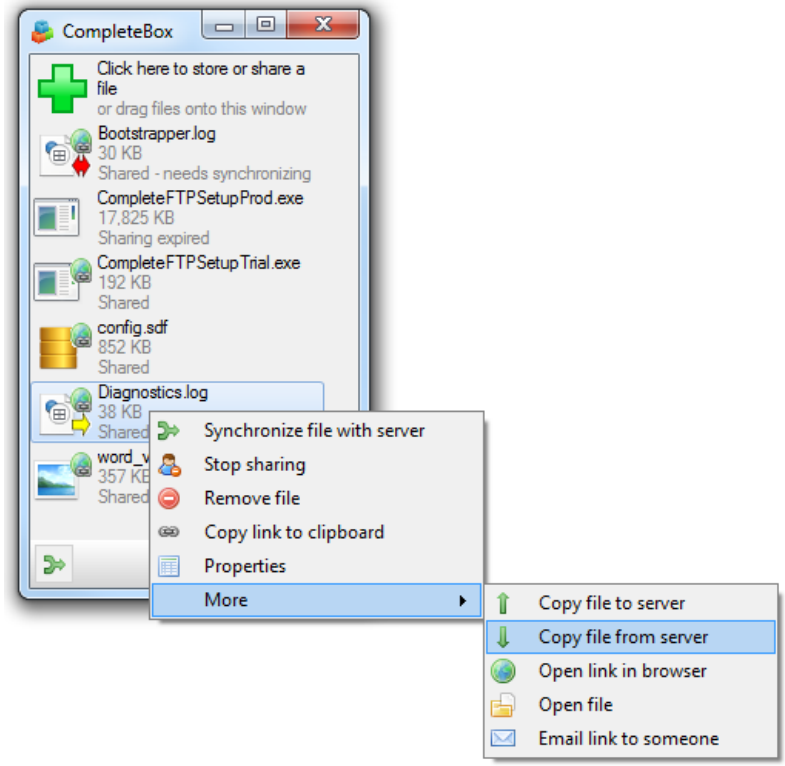
**How to synchronize files**

To synchronize the local and server files, right-click on the filename and use the *Synchronize file with server* menu item, as shown below. This will ensure the latest copy is the local and the remote file - it works via the timestamp.

You can also synchronize **all** out-of-date files via the green synchronize button that appears on the bottom left of the CompleteBox file listing, shown on the left.

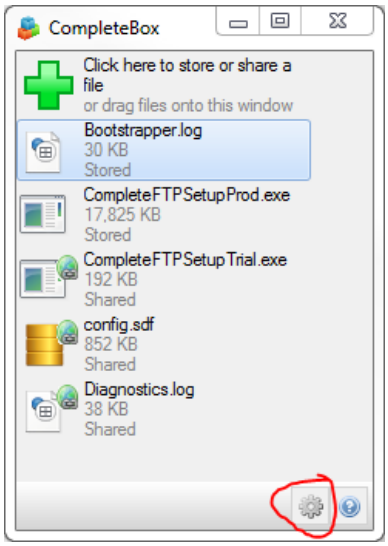
**Manually overriding synchronization**

Occasionally, you may find that you want to overwrite local changes that you have not yet synchronized to the server. For example, you may have made some local changes, and wish to discard them. This can be done by right-clicking on the filename and choosing *More -> Copy file from server*. This copies the server file over the local file, erasing the local changes. The *Copy file to server menu item* does the oppposite, forcing the local file to be copied to the server, overwriting any changes in the server file.

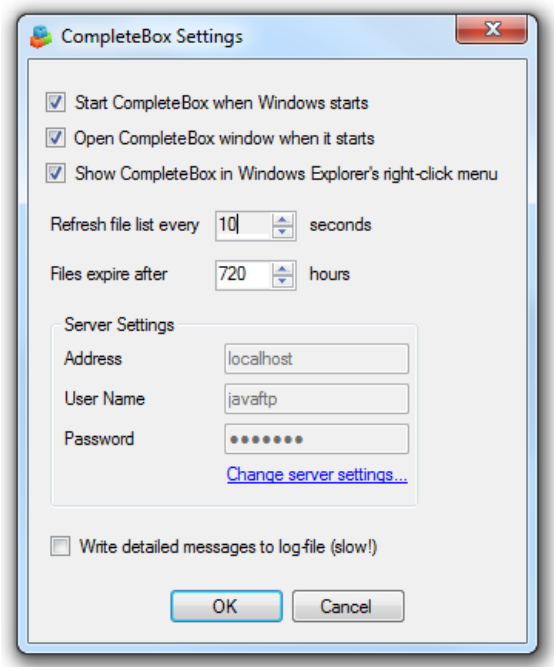


## How to configure CompleteBox

*CompleteBox's* configuration dialog is found by selecting the cog icon on the main dialog that displays the list of shares, shown on the left.




Clicking the cog icon shows the following dialog:




There are various options controlling if CompleteBox is started with Windows, as well as enabling or disabling the Windows Explorer integration.

The CompleteBox client automatically refreshes its shared files listing every 10 seconds from the server while the window is active, and this can be altered. The default share expiry time of 720 hours (30 days) can also be changed here.

The credentials can also be changed in this dialog - both the username and password, and the sharing server that is being used.

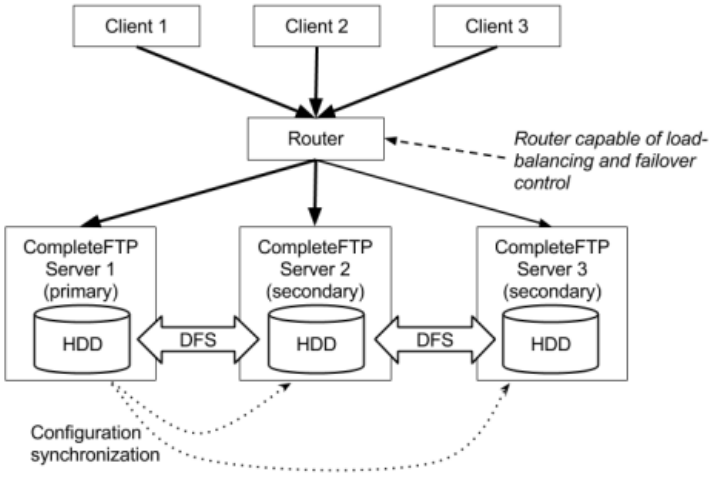
Finally, if any problems are experienced that require contacting support, detailed logging can be switched on, and the zipped log file emailed to support. Note that this slows down file transfers considerably, and should only be used temporarily. The log file can be found at

`C:\Users\username\AppData\Local\EnterpriseDT\CompleteBox\CompleteBox.log`.

## Clustering

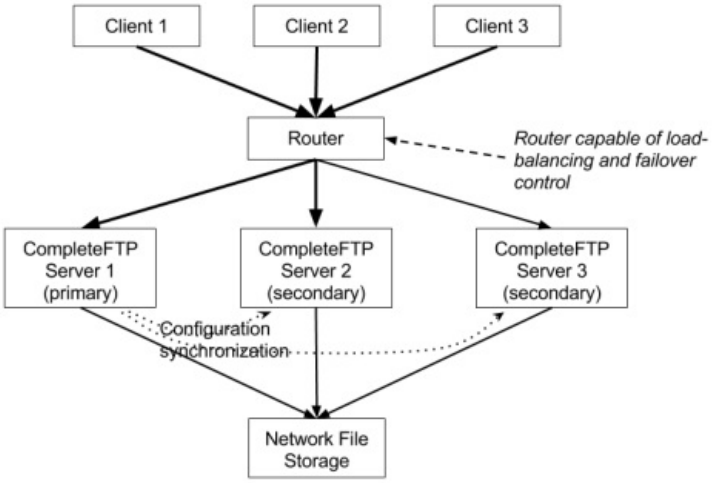
The Enterprise Edition of CompleteFTP provides support for *clustering* of multiple servers across different machines (from 7.0). Native clustering is not supplied, but in combination with a hardware or software load balancer (e.g. Microsoft NLB) load balancing and failover can be achieved.

What CompleteFTP Enterprise supplies is synchronized and instantaneous server configurations across the cluster. A server instance is designated as the *primary* server, and configuration changes made on the primary are instantly propagated to all the *secondary* servers in the cluster. Hence the most suitable architecture is an active/active cluster setup - in an active/passive cluster, configuration changes cannot be propagated to the dormant machine. A single instance of the CompleteFTP Manager can control all the servers in the cluster. An architecture diagram is shown below:



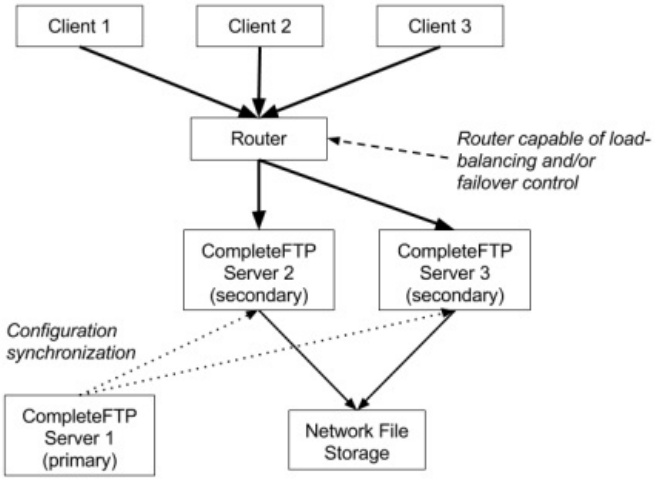

The diagram shows three clients connecting to a cluster via a router. This could be be done via any supported protocol, such as FTPS, SFTP or HTTPS. The router would be capable of load-balancing and/or failover control. It directs incoming requests to one of the three servers. The way in which it selects the server depends on how it's configured. The three servers are all running CompleteFTP and are configured to all be capable of responding to the same requests. One of the servers is set to be the primary from within CompleteFTP. All configuration changes must be made via this server. It is responsible for distributing configuration changes to the secondaries. This is done in near-real-time. While CompleteFTP takes responsibility for synchronizing configuration changes between the three servers, it does not synchronize the files on the Hard Disk-Drive (HDD) - this is left to Windows DFS. The Windows DFS (Distributed File-System) is a feature of Windows Server 2008/2012 that synchronizes files and directories between server instances in near-real-time.

The diagram above shows an architecture where user's files are store locally on each server. Below is an architecture where files are on network storage:




DFS is now not required on the CompleteFTP servers since no user-files need to be synchronized between them.

In situations where load is very high it may be preferable to minimize load on the primary node. In this case the following architecture could be used:




The main motivation for this is to reduce the load on the primary node. The primary node is more important than the secondaries since it's a central point for synchronization of configuration data for the following reasons: it's the node to which the administration console (CompleteFTP Manager) must connect in order to make configuration changes it is responsible for distributing those changes to the secondaries it is the node from which a secondary that is booting up will request a configuration update. While this architecture is preferable to the other architectures, it is by no means essential - it is only marginally better.

One important caveat relates to HTTP/S. As each HTTP/S session is local to each CompleteFTP server, the load balancer will need to ensure all connections from the same external IP address are directed to the same CompleteFTP instance in the cluster. A future release will replicate HTTP/S session caches across the cluster, alleviating this requirement for the load balancer.

# How to enable clustering

The Enterprise Edition of CompleteFTP supports *clustering* as described in detail here.

A CompleteFTP cluster consists of a *primary* server and one or more *secondary* servers. All configuration changes are made on the primary and propagated to the secondaries within seconds.

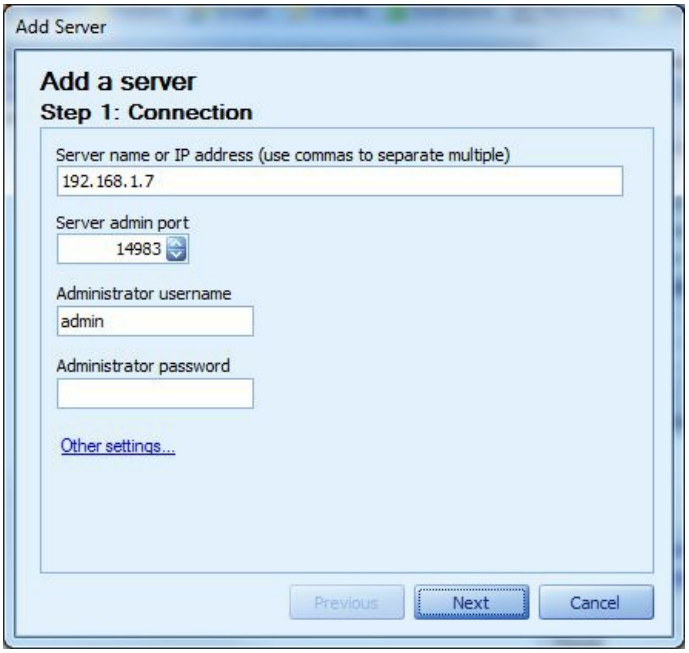
To set up a cluster, CompleteFTP needs to be installed and running on the machine designated as the primary, and also on the machines that will host secondary servers. It must also be possible for the primary server to be able to communicate with the secondary servers across the network via:

1. the protocol that is **initially** set for the *admin* user on each secondary, and
2. the protocol that will be used by **all** servers after they have been added to the cluster

It is important that firewall rules are configured to allow this server to server communication. By default, the SFTP protocol is used, which requires communication on port 14983.

To designate a server as a primary server, the CompleteFTP manager must be connected to that server. In the *Overview* panel, the *Servers* window is used to control the servers. Select *Add server* to add another server as a secondary - this will automatically designate the current server that the CompleteFTP manager is connected to as the primary.

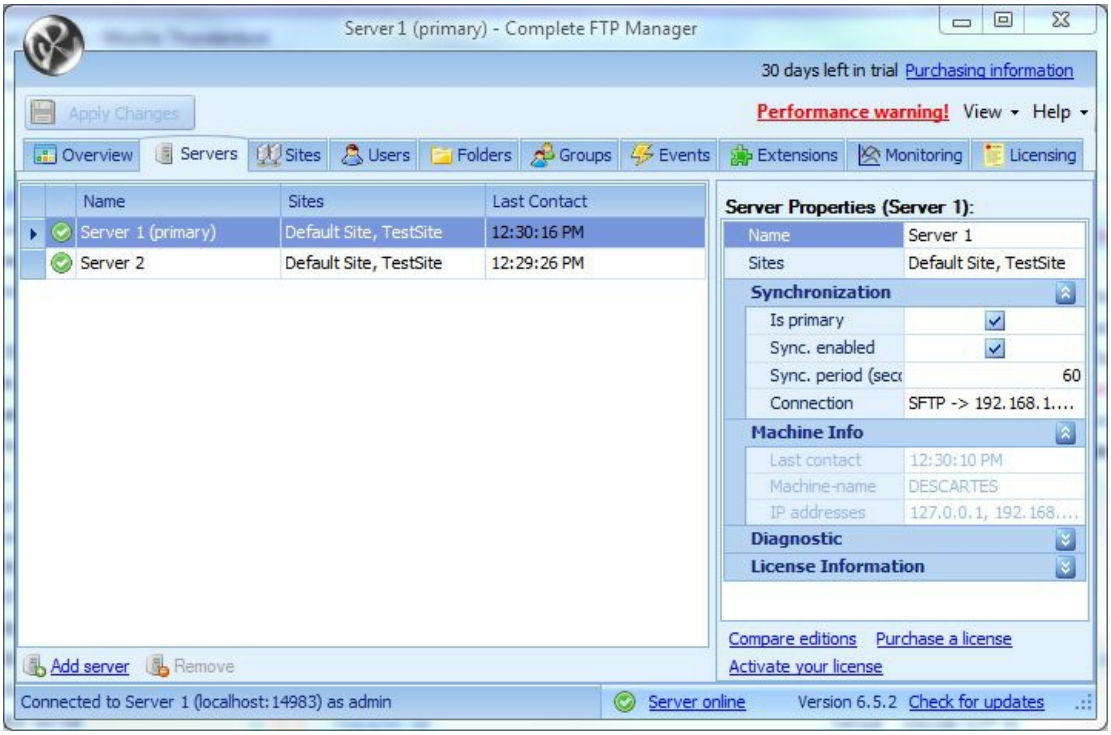
*Add server* brings up the dialog box below:




Enter the details of the secondary server, and select the Next button. In the follow dialog, select the Add button to add the secondary server to the cluster. A progress report will be produced, detailing the registration process. As part of this process, the secondary server has its configuration overwritten with a copy of the primary server's configuration.

If the registration process is successful, the new server will now be a secondary server in the cluster that is controlled by the primary. All future configuration changes on the primary will be instantly propagated to the secondary as soon as they are applied (provided the "Sync enabled" checkbox is checked, which means automatic synchronization of changes). Also note that the frequency of synchronization can be altered from the default 60 seconds.

Once a server has been added to create a cluster, the servers will be listed in the Servers tab, together with the sites they support and the last time they were contacted (i.e. synchronized with the primary's configuration).The end result should look like the image below:



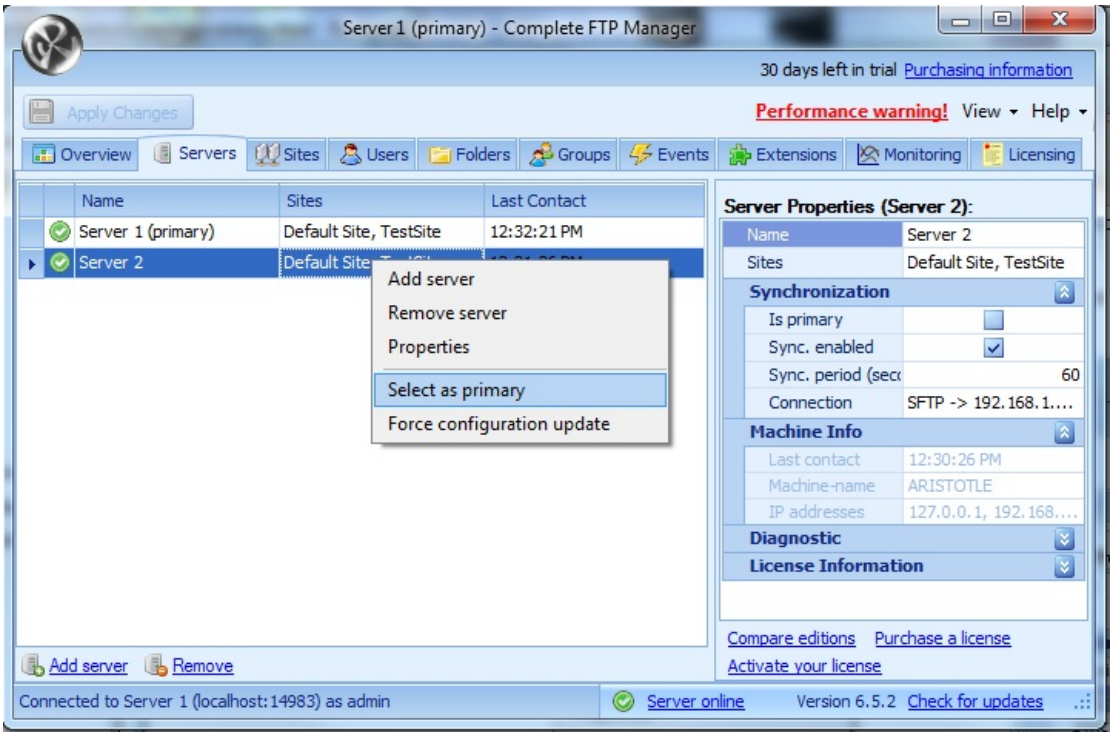

Additional servers can be added by using the "Add server" link at the bottom left of the Servers tab, or by using "Add server" from the menu that is shown when the right mouse button is clicked. Servers can also be removed from the cluster by selecting "Remove server".

# How to change primaries

Sometimes it is necessary to choose a different machine in the cluster to be the primary server. Perhaps the current primary needs replacing or upgrading, for example.

If the new primary-to-be is not yet in the cluster, first add it to the cluster as a secondary server.

To select a secondary server in the cluster as the new primary, highlight it in the server list, and use the right mouse button click menu to choose "Select as primary". This menu is shown below.




Finally, if the old primary is no longer to be in use, remove it from the cluster (using "Remove Server" from the new primary's menu), and then decommission it.

# How to resolve clustering problems

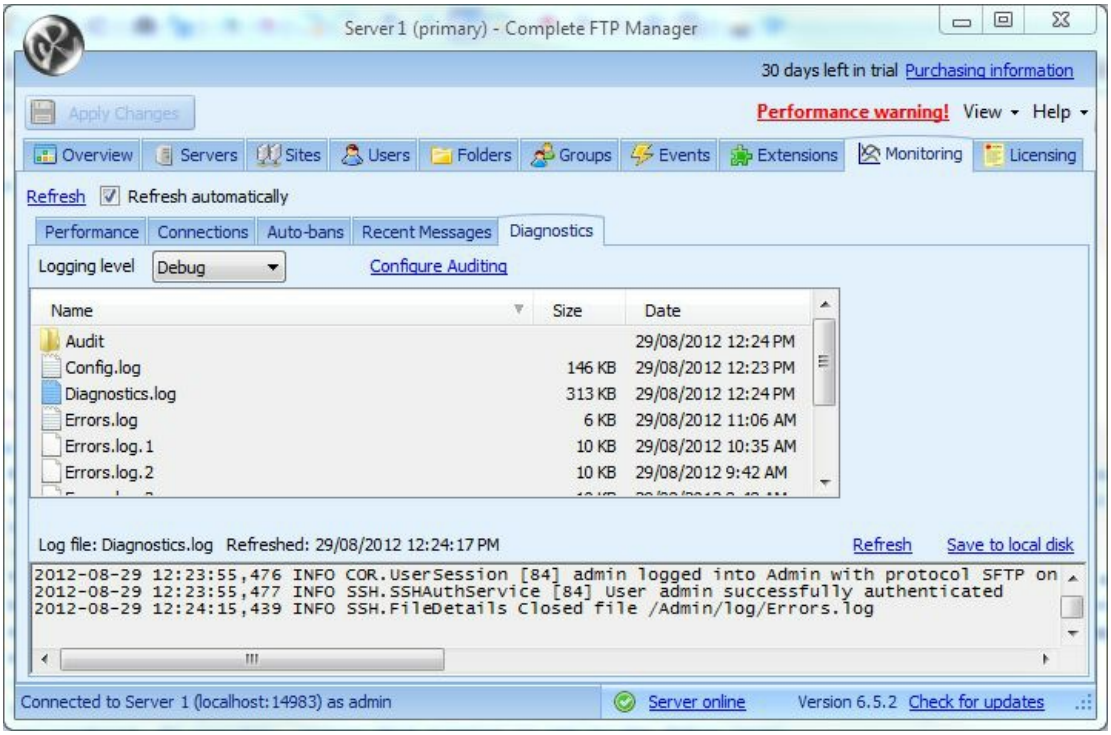
Running a group of servers in a cluster is considerably more complicated than a single server, particularly when first configuring the cluster. This is because configuration changes are being distributed to all servers in the cluster.

**Forcing configuration updates**

There are two simple things that can be tried to resolve any problems. If a secondary is out of date and cannot be synchronized, the right mouse button click menu item"Force configuration update" can be used to force a secondary server to have its configuration overwritten using the configuration of the primary. An alternative approach is to simply remove the server from the cluster and re-add it.

**Logging**

If problems are experienced that cannot be resolved by the above, it will be necessary to look at the log files that are produced. If possible, ensure that logging is set to the Debug level, as shown below:


Server 1 (primary) - Complete FTP Manager
30 days left in trial Purchasing information
Apply Changes
Performance warning! View Help
Overview Servers Sites Users Folders Groups Events Extensions Monitoring Licensing
Refresh Refresh automatically
Performance Connections Auto-bans Recent Messages Diagnostics
Logging level Debug
Configure Auditing
Name Size Date
Audit 29/08/2012 12:24 PM
Config.log 146 KB 29/08/2012 12:23 PM
Diagnostics.log 313 KB 29/08/2012 12:24 PM
Errors.log 6 KB 29/08/2012 11:06 AM
Errors.log.1 10 KB 29/08/2012 10:35 AM
Errors.log.2 10 KB 29/08/2012 9:42 AM
Log file: Diagnostics.log Refreshed: 29/08/2012 12:24:17 PM
Refresh
Save to local disk
2012-08-29 12:23:55,476 INFO COR.UserSession [84] admin logged into Admin with protocol SFTP on
2012-08-29 12:23:55,477 INFO SSH.SSHAuthService [84] User admin successfully authenticated
2012-08-29 12:24:15,439 INFO SSH.FileDetails Closed file /Admin/log/Errors.log
Connected to Server 1 (localhost:14983) as admin
Server online
Version 6.5.2
Check for updates

Ensure that **all** log files from **every** server in the cluster are sent to us. Please ensure that they are zipped and that it is clear which machine each group of logs comes from. A simple way to do this is to copy each group of files to a directory that has the name of the server, and zip up all these directories into the one zip file.

# How to activate cluster nodes

A CompleteFTP cluster consists of a *primary* server and one or more *secondary* servers. The primary server adds secondary servers to the cluster as described here.

Typically, each server in the cluster (including the primary) must be activated separately, to obtain and set its own activation key.

From version 7.2, there is another way to accomplish activation on clusters for Corporate License customers, using their universal activation key (UAK). If the UAK is installed first on the primary server, then it will be automatically propagated to each secondary server as it is added to the cluster. This means secondary servers can be automatically activated from the trial state.

# How to upgrade a cluster

Upgrading is usually quick and trouble-free and all your configuration data is maintained (as well as being backed up prior to updating).

Please note that all nodes in the cluster must be the same version. Please ensure you download the production installation - the link (including credentials) is included in your purchase email. The trial installation will not install.

Do the following steps in order:

1. Close CompleteFTP Manager on the primary.
2. Stop the CompleteFTP service on the primary.
3. On each of the secondaries:
    - Close CompleteFTP Manager if it's open.
    - Uninstall the old version of CompleteFTP.
    - Install the new version of CompleteFTP.
    - Take care to enter the same administrator password on all the nodes.
4. Once you've upgraded each of the secondaries do the same to the primary, again making sure to enter the same administrator password.

The CompleteFTP service will be started on each secondary machine during installation and will keep trying to contact the primary, which will of course fail until it's been upgraded and restarted. Once the primary has been upgraded it will restart and immediately contact each of the secondaries. As long as the passwords all match the cluster should now be back online and synchronized again.

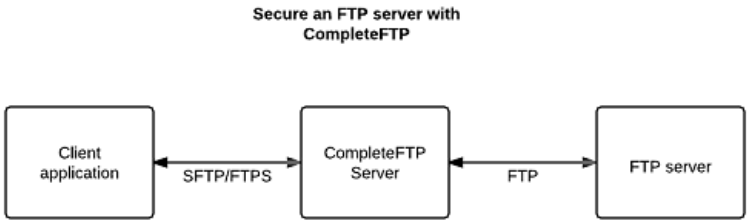
# Gateway

CompleteFTP (Enterprise Edition only) can act as a *multiprotocol gateway* for other servers. It can present any of its own protocols to a client, and translate them into a different protocol to talk to another server.

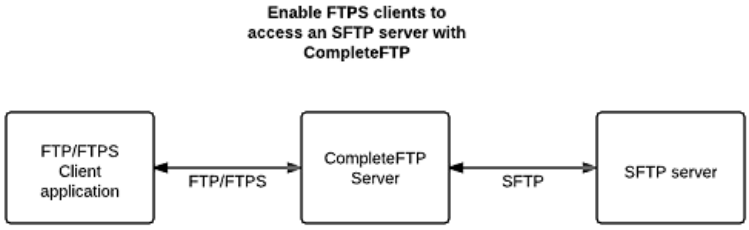
The multiprotocol gateway is an extremely powerful feature. A number of possible uses are described below. Gateway configuration is described here.

**Gateway examples**

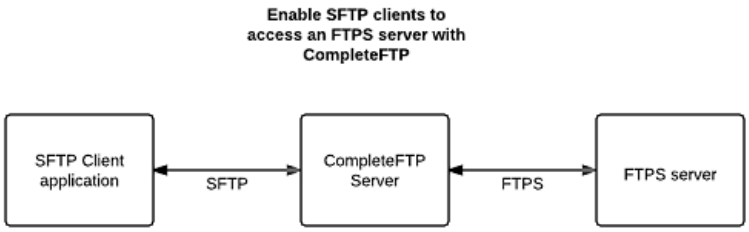
1. Provide secure access to another non-secure FTP server located on the same machine or network. CompleteFTP provides a secure wrapper to the secure server. This means SFTP or FTPS clients can connect to CompleteFTP, which proxies the requests on to the non-secure server.



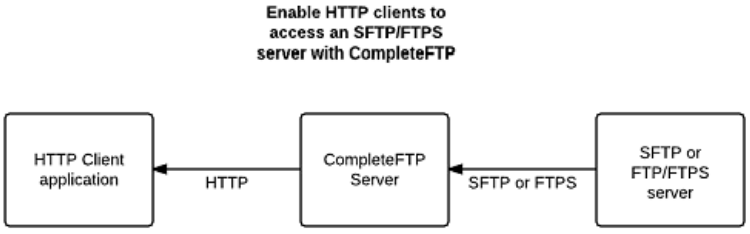

2. Add FTP or FTPS support to an existing SFTP server. This means FTP or FTPS clients can access the SFTP server without any other changes than setting up CompleteFTP to proxy the requests.




3. Add SFTP support to an existing FTP or FTPS server. This means SFTP clients can access the FTP or FTPS server without any other changes than setting up CompleteFTP to proxy the requests.




4. Add HTTP support to an existing FTP, FTPS or FTPS server. This means SFTP clients can access the FTP or FTPS server without any other changes than setting up CompleteFTP to proxy the requests.



## Gateway configuration

There are two ways of configuring the gateway to access a particular server. Both cases require a gateway folder to be configured, which is analogous to a remotely mounted directory. When users login and navigate to a gateway folder, they are actually accessing the remote file system. A gateway folder can be set as a user's home directory.

To configure the gateway to access a remote server, either:

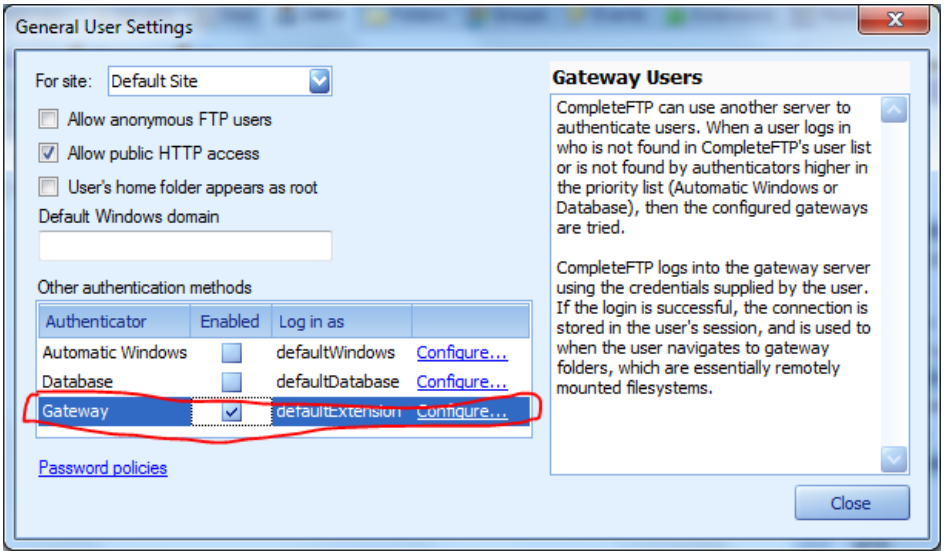
1. Set up a gateway authenticator, where the remote server is used to authenticate a user that is trying to log in. The authenticated connection to the server is saved in the user's session. Then set up the gateway folder to use the gateway connection, or
2. Set up a gateway folder that uses a specified connection, i.e. the gateway folder has the proxied server's credentials added, and every user that accesses the gateway folder uses these credentials.

# How to configure a gateway authenticator

Gateway users require a gateway authenticator to be configured to allow users to be authenticated using a different server. This is a feature of the Enterprise Edition only of CompleteFTP.

A gateway authenticator is an authentication extension that authenticates user attempting to log in via another server, using whatever protocol is selected. When the user details are received in CompleteFTP, a login attempt to the remote server is attempted using the supplied credentials.

To configure a gateway authenticator, select "General user settings" link from the Users tab. "Gateway" is listed amongst "Other authentication methods", and has a "Configure" link that must be selected (below).


General User Settings
For site:
Default Site
Allow anonymous FTP users
Allow public HTTP access
User's home folder appears as root
Default Windows domain
Other authentication methods
Authenticator
Enabled
Log in as
Automatic Windows
defaultWindows
Configure...
Database
defaultDatabase
Configure...
Gateway
defaultExtension
Configure...
Password policies
Gateway Users
CompleteFTP can use another server to authenticate users. When a user logs in who is not found in CompleteFTP's user list or is not found by authenticators higher in the priority list (Automatic Windows or Database), then the configured gateways are tried.
CompleteFTP logs into the gateway server using the credentials supplied by the user. If the login is successful, the connection is stored in the user's session, and is used to when the user navigates to gateway folders, which are essentially remotely mounted filesystems.
Close

This brings up the "Gateway configuration" dialog box which allows the addition and removal of gateways. Multiple gateways can be configured, meaning multiple different machines can be tried for authentication, in the order in which they are listed.

There are both "Basic" and "Advanced" connection properties to be configured for each gateway. Usually the basic properties are all that is required - the protocol, the server address and port, and optionally the initial directory to change to on the remote machine.

If the basic settings are not sufficient, the advanced settings should be examined. These include character encoding; connection settings such as proxies, timeouts and active and passive settings; directory listing settings such as caching of listings; and transfer settings. These settings permit a fine degree of configuration control over a gateway authenticator.

# Groups

The *Groups* panel is used for displaying and controlling the access permissions of groups of users to system resources, usually folders.

This panel allows you to create new groups and add users to those groups. Once groups have been created as desired, permissions may be assigned to those groups.

Group permissions for a given folder may be set by going to the File System panel; selecting the folder; setting the Group of the folder; clicking on the Permission property and checking the checkboxes in the group section of the permission editor.

# Events

In CompleteFTP, an *event* is an action performed by a logged in user on the server.

For example, uploading or downloading a file is an event.

Email notifications can be attached to these events, so that when, for example, a file is uploaded, an email is sent to a specified user providing details of the upload. Email notifications are described in more detail here.

Process triggers can also be attached to these events, so that when the event is triggered, a program can be launched. Process triggers are described in more detail here.

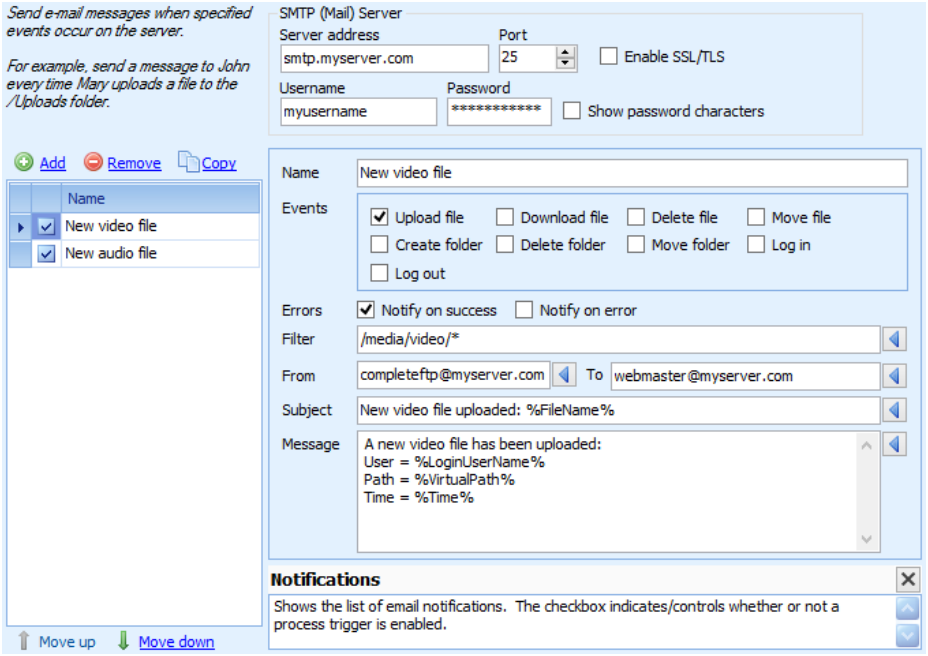
Scheduled events can be defined to trigger using a predefined schedule, such as once per hour or at midnight every weekday. See here for details.

These events can also be recorded in the audit log.

Note that email notifications, process triggers and auditing are not supported in the Standard Edition.

# How to set up email notifications

CompleteFTP supports highly customizable email notifications. This means that when an event such as a file upload occurs, an email can be sent.




To set up email notifications, the mail server (SMTP server) settings must be entered. The server hostname, port number (normally port 25) and optional username and password are entered in the Email Notification tab found from the *Events* panel. SSL connections are also permitted by checking the "Enabled SSL/TLS" checkbox. For SSL, the port number will most likely need to be changed - the default SMTP SSL port is 465.

The list of current notifications is displayed in the Email Notification tab. Use *Remove* to delete notifications, and *Add* to create a new notification.

Enter an appropriate name for the notification, and select what Events the notification is applicable for. The various fields are described below.

**Events**

Processes may be triggered by any of the types of events listed below. By default events are only triggered when an operation, such as a file upload, succeeds, but they may also be triggered on failure. This is determined by the checkboxes labelled 'Trigger on success' and 'Trigger on error'.

**Upload file**
When a file has been uploaded.

**Download file**
When a file has been downloaded.

**Delete file**
When a file has been deleted.

**Move file**
When a file has been renamed or moved.

**Create folder**
When a new folder has been created by a user.

**Delete folder**
When a folder has been deleted by a user.

**Move folder**
When a folder has been renamed or moved by a user.

**Log in**
When a user logs in.

**Log out**
When a user logs out.

**Filters**

The folder- or file-path filter should be entered. This filter may be used to select the files/folders which the notifications should be sent for. For example, if notifications should only be sent for files with the extension, '.dat', then the filter should be set to '*.dat'. By default Windows-style wildcards are used for matching.

Note that the full path is used for matching; not just the file-name. So if notifications should be sent for files named 'test.dat' in any directory, then the filter should be set to '*/test.dat'.

Regular expressions .NET-style) may also be used. To use these the filter should be prefixed by 'regex:'. Information on .NET regular expressions may be found on the Microsoft(tm) website. Note that multiple filters may be defined using regular expressions using the syntax, `regex:^(/Folder1/.*|/Folder2/.*)`, which would match anything

underneath the folders, `/Folder1` and `/Folder2`. If you want to match anything **not** underneath these folders you can do that using the regular expression, `regex:^(?!/Folder1/.*|/Folder2/.*)`.

**From address**

Enter the email address that the email notification is set to be from. The *%OwnerEmail%* (email address of folder owner) or *%LoginEmail%* (email address of logged-in user) macros may be used here.

**To address**

Enter the email address that the email notification is sent to. The *%OwnerEmail%* (email address of folder owner) or *%LoginEmail%* (email address of logged-in user) macros may be used here.

**Subject**

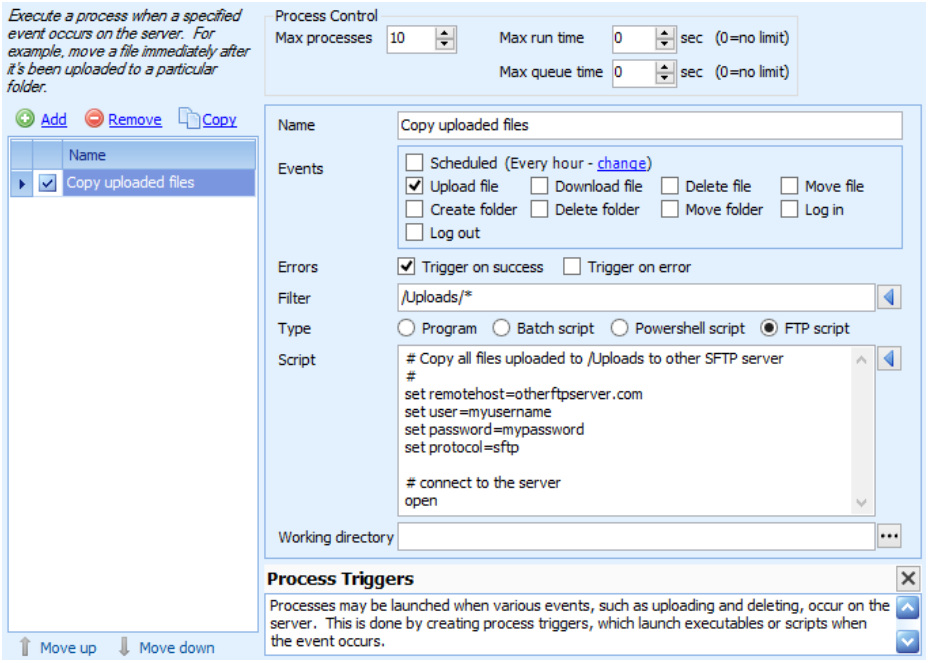
Enter the subject of the notification message. Various macros can be used here.

**Message**

Enter the body of the notification message. Various macros can be used here.

# How to set up process execution triggers

CompleteFTP supports highly customizable process execution triggers. This means that when an event such as a file upload occurs, a process can be launched (e.g to move a file after it has been uploaded).




**Process control**

In the Process Execution tab, the maximum number of concurrent processes can be set (default 10), together with the maximum run time and maximum time a process is queued for before it runs. The latter two setting have a default of 0 which means no limit.

**Creating and editing triggers**

The list of current process triggers is displayed in the Process Execution tab. Use *Remove* to delete triggers, and *Add* to create a new trigger.

When adding, enter an appropriate name for the trigger, and select what Events the trigger is applicable for. The various fields are described below.

**Events**

Processes may be triggered by any of the types of events listed below. By default events are only triggered when an operation, such as a file upload, succeeds, but they may also be triggered on failure. This is determined by the checkboxes labelled 'Trigger on success' and 'Trigger on error'.

**Scheduled**
Events occur at a predefined schedule, such as once per hour or at midnight every weekday. See here for details. Since this event is not related to any particular session macros related to users and files are not available.

**Upload file**
When a file has been uploaded. All macros are available.

**Download file**
When a file has been downloaded. All macros are available.

**Delete file**
When a file has been deleted. All macros are available.

**Move file**
When a file has been renamed or moved. All macros are available.

**Create folder**
When a new folder has been created by a user. All macros are available.

**Delete folder**
When a folder has been deleted by a user. All macros are available.

**Move folder**
When a folder has been renamed or moved by a user. All macros are available.

**Log in**
When a user logs in. All macros are available.

**Log out**
When a user logs out. All macros are available.

**Filters**

The folder- or file-path filter should be entered. This filter may be used to select the files/folders which the notifications should be sent for. For example, if notifications should only be sent for files with the extension, '.dat', then the filter should

be set to '*.dat'. By default Windows-style wildcards are used for matching.

Note that the full path is used for matching; not just the file-name. So if notifications should be sent for files named 'test.dat' in any directory, then the filter should be set to '*/test.dat'.

Regular expressions (.NET-style) may also be used. To use these the filter should be prefixed by 'regex:'. Information on .NET regular expressions may be found on the Microsoft(tm) website. Note that multiple filters may be defined using regular expressions using the syntax, `regex:^(/Folder1/.*|/Folder2/.*)`, which would match anything underneath the folders, `/Folder1` and `/Folder2`. If you want to match anything **not** underneath these folders you can do that using the regular expression, `regex:^(?!/Folder1/.*|/Folder2/.*)`.

**Type**

Three types of process triggers can be created. They can either be a normal program (an executable file such as cmd.exe), a DOS batch script or a Powershell script. The settings for these triggers are described below:

**Program file**

Only applicable for Programs. This is the path to the executable that will be launched. Note that a program with a GUI should NOT be specified.

**Arguments**

Enter the arguments to be supplied to the program. Various macros can be used here.

**Script**

Enter the batch script, Powershell or FTP script to be run. Various macros can be inserted into the script at execute time.

**Working directory**

Select the working directory that the program or script should be running in when it is executed.

**Examples**

For both the examples below, if they are to apply to all uploaded files, use "*" as the filter. Otherwise use a specific directory, e.g. for a certain user, /home/username/*. Note that in the scripts the Windows paths must be used rather than the virtual filesystem paths.

To move all uploaded files to a single directory after upload (as a batch file):

```
move "%WindowsPath%" D:\destination
```

To change files to readonly after uploading:

```
attrib +r "%WindowsPath%"
```

# Scheduled Events

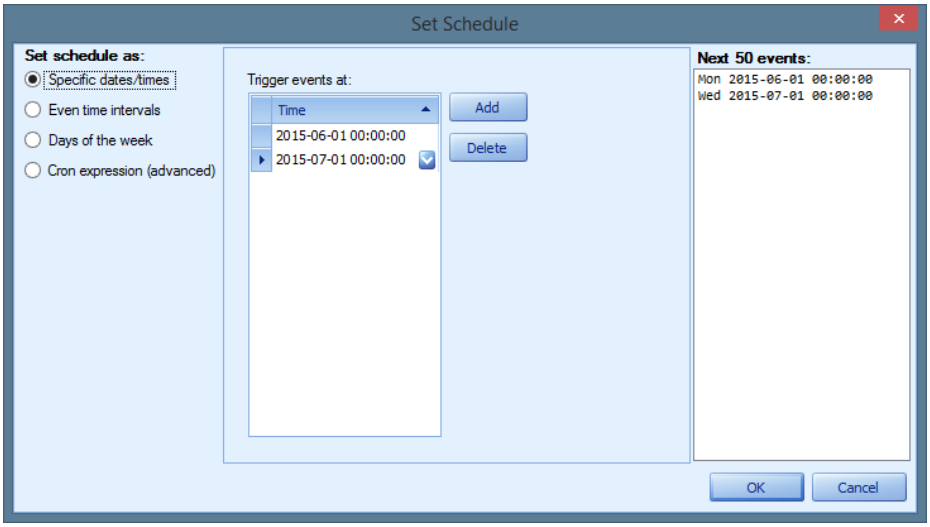
CompleteFTP allows events to be triggered according to a user-specified schedule. This schedule may consist of a sequence of specific times or of a repeated pattern. There are three ways to define a pattern: by even time intervals, by days of the week or by a cron expression.

Schedules may be defined using the Set Schedule dialog. This is accessed on the Process Triggers panel by selecting the *Change* link next to the "Scheduled" checkbox of the event.

The four methods of setting a schedule are listed on the left. When one of these is selected the editor for that method will be displayed in the centre. On the right is a list of the next (up to) 50 times that the defined event will be triggered.

**Specific times**

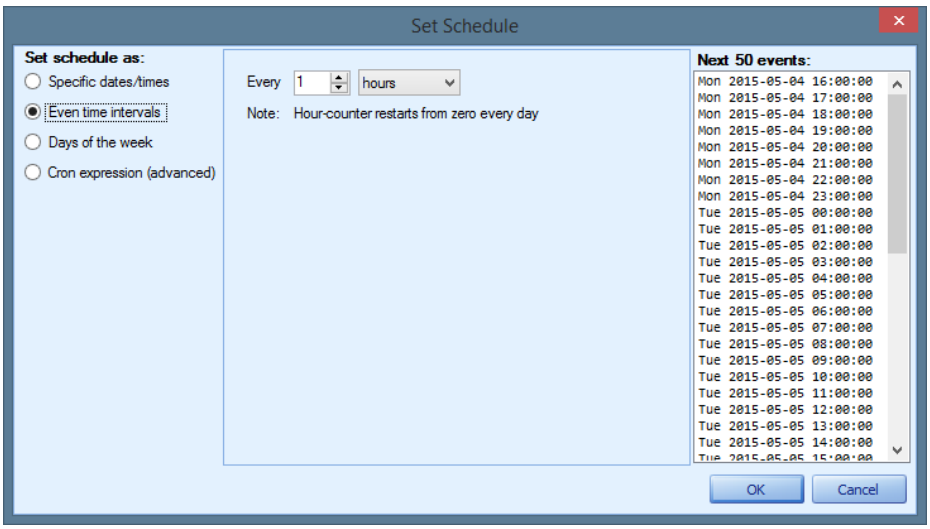
A schedule of specific times is defined by creating a list of dates and times using the following interface:




The 'Next 50 events' list will only list future events and a warning will be shown when OK is pressed if there are no future events.

**Even time intervals**

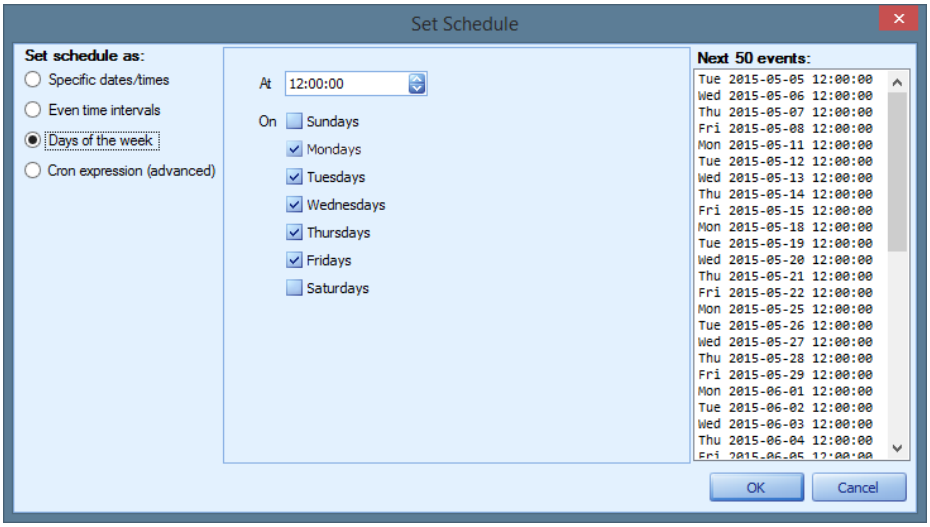
This is the easiest way to define an event that is to be trigged every *n* seconds, minutes, hours, days or months.




It's important to note that the counter restarts from zero when it passes the maximum value of the particular unit that is selected. For example, if 23 minutes is selected and the time is currently 13:11 then the event will trigger at 13:23, then 13:46 and then again at 14:00. It's therefore best to use factors of the maximum value, such as 30 seconds, 15 minutes or 6 hours.

**Days of the week**

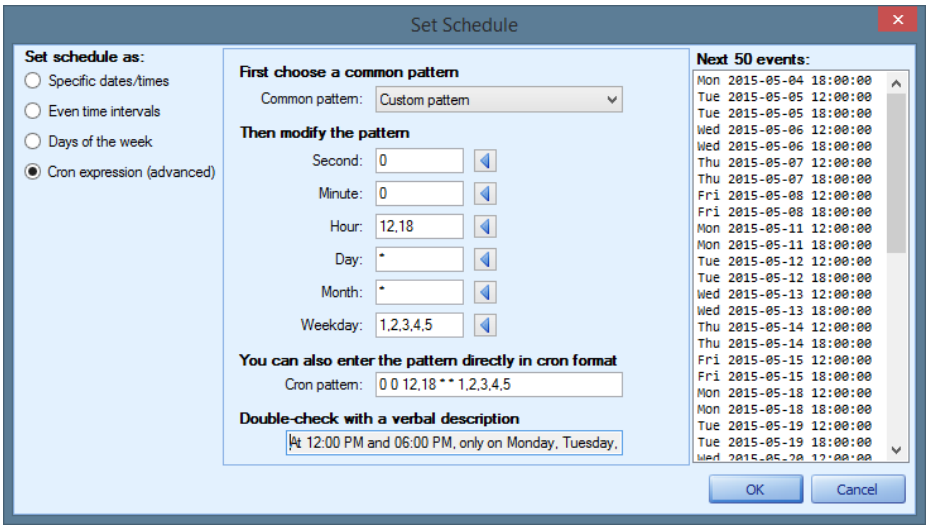
Events may be specified to trigger at a certain time on specific days of the week. For example, at noon on weekdays:




## Cron expression (advanced)

If none of the simpler methods of defining schedules cover your requirements then cron expressions probably will. Cron expressions are a powerful concept often used in Unix-like operating systems. Please refer to this Wikipedia article for an explanation. Note however that CompleteFTP's cron expressions also includes seconds.

The interface for specifying cron expressions allows you to select a common pattern from the list at the top. If this doesn't meet your requirements then it may be customized by modifying each of the components in turn. The button to the right of each component text-field allows you to select from a set of common values for that field. You may also edit the cron expression directly in the text-field labelled 'Cron pattern'. The field at the bottom provides a verbal description of the current cron expression.



# How to write FTP scripts

FTPScript is a powerful yet simple scripting language specifically for FTP, FTPS and SFTP.

**Command overview**

Most commands are fairly self explanatory, and are described in the script command reference.

The set command is used to set various parameters that are used during the session, such as username, password, timeout and so on. Many of these parameters are optional (e.g. port, timeout). The most important parameter is the protocol.

Various macros can be inserted into the script at execute time.

The local working directory may be set in the process trigger definition inside CompleteFTP Manager, or it may be set using the localdir command inside the script. All uploads and downloads that do not specify a full path will use the set local working directory.

**Sample code**

A sample script is shown below:

```
# Sample SFTP script for uploading a file to another server
#
set remotehost=otherftpserver.com
set user=myusername
set password=mypassword
set protocol=sftp

# connect to the server
open

# upload the file for which the trigger occurred to the server
put %WindowsPath% %FileName%

# disconnect
close
```

## Script command reference

Scripting and shell commands are described in the table below. Abbreviations for commands are noted in the command description.

Where the command has optional arguments, they are shown in square brackets, e.g. *open [host[:port]]* means that for the open command, the host and port are optional parameters (but if the port is supplied, the host must be supplied).

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>help [command]</td><td>Prints out help for a specific command, or if no command is specified, a list of supported commands is provided.</td></tr>
<tr><td>version</td><td>Prints out version details.<br><br>Can use ver as an abbreviation.</td></tr>
<tr><td>debug</td><td>Switches debug on or off. If debug is already on, it is switched off (and vice versa). When debug is enabled, more information is displayed when commands are executed.</td></tr>
<tr><td>set</td><td>Sets various parameters used during the session, many of which are optional. Syntax is <code>set name=value</code><br><br>Note that 'set' can be omitted, and thus <code>protocol=ftp</code> is a valid command.<br><br>Variables can be created this way. If an unknown variable is set using the 'set' command, then a variable of that name and value is created, e.g. myprotocol=ftp creates a variable called 'myprotocol' set to the string value 'ftp'.<br><br>The value of variables can be printed out using echo.<br><br>The value of all set variables can be printed out using set without any arguments.<br><br>The set command also supports a feature originally found on Unix systems called backquoting. This is where a variable is assigned the value of the output of a command, which must be surrounded by back quotes (the "`" symbol).<br><br>For example, to set the variable 'currendir' to the value of the current remote working directory, use <code>set currentdir=`pwd`</code>.</td></tr>
<tr><td>set connectmode</td><td>Sets the connection mode for data transfers and listings. Not applicable for SFTP protocol. Possible values are active or passive., e.g. <code>set connectmode=active</code>.</td></tr>
<tr><td>set connectmode</td><td>Sets the connection mode for data transfers and listings. Not applicable for SFTP protocol. Possible values are active or passive., e.g. <code>set connectmode=active</code>.</td></tr>
<tr><td>set protocol</td><td>Set the protocol to be used. Note that the protocol cannot be changed while connected. Supported protocols are shown below:<br><br>
<table>
<tr><th>Protocol</th><th>Description</th></tr>
<tr><td>ftp</td><td>"Plain" FTP protocol</td></tr>
<tr><td>ftps</td><td>FTPS, i.e. FTP over SSL (explicit mode)</td></tr>
<tr><td>ftpse</td><td>FTPS, i.e. FTP over SSL (explicit mode)</td></tr>
<tr><td>ftpsi</td><td>FTPS, i.e. FTP over SSL (implicit mode).</td></tr>
<tr><td>sftp</td><td>SFTP, i.e. FTP over SSH</td></tr>
</table>
<br>Note that server validation is switched off by default.</td></tr>
<tr><td>set connectmode</td><td>Sets the connection mode for data transfers and listings. Not applicable for SFTP protocol. Possible values are active or passive., e.g. <code>set connectmode=active</code>.</td></tr>
<tr><td>set remotehost</td><td>Sets the remote host to be connected to, e.g. set remotehost=ftp.gnu.org</td></tr>
<tr><td>set user</td><td>Sets the username, e.g. <code>set user=javaftp</code>. This username is used for all supported protocols.</td></tr>
<tr><td>set password</td><td>Sets the user's password, e.g. <code>set password=javaftp</code>. This password is used for all supported protocols.</td></tr>
<tr><td>set localdir</td><td>Sets the local working directory. All uploads and downloads that do not specify a full path will use this directory. e.g. <code>set localdir=D:\work\tmp</code></td></tr>
<tr><td>set servervalidation</td><td>Enables or disables server validation (disabled by default). Possible values are true or false.</td></tr>
<tr><td>set clientcert</td><td>Sets the path of the client certificate to use for client validation for FTPS. Currently supports PEM, CER and PFX file formats for certificates. Note that the password must also be set via certpassword.<br><br>Once set, client validation is automatically switched on.</td></tr>
</table>

| | |
|---|---|
| **set certpassword** | Sets the password (or passphrase) of the client certificate used for client validation (FTPS). |
| **set serverpublickey** | Sets the path of the server's public key to use for server validation for SFTP. Supports OpenSSH and SECSH file formats. More typically `set knownhosts` is used.<br><br>Can also use *serverkey*.<br><br>Server validation must also be switched on via `set servervalidation=true` |
| **set knownhosts** | Sets the path of the *known_hosts* file to use for server validation for SFTP.<br><br>Server validation must also be switched on via `set servervalidation=true` |
| **set clientprivatekey** | Sets the path of the client private key file to be used for public key authentication in SFTP. Supports PuTTY, OpenSSH and SSH.com private key formats. Note that the *clientprivatekeypass* must also be set.<br><br>Can also use *clientkey*. |
| **set clientprivatekeypass** | Sets the passphrase of the client private key file to be used for public key authentication in SFTP.<br><br>Can also use *clientkeypass*. |
| **set timeout** | Optional. Sets the timeout in seconds for read or write operations. The default is 0 (which is an infinite timeout). e.g. set timeout=10 |
| **set port** | Optional. Defaults are determined by the protocol. Sets the port number to connect to, e.g. set port=21 |
| **set dataports** | Optional. Only applicable for active mode. Sets the data port range that the client will listen on for listings and transfers. The low port number must be listed first, e.g. set dataports=10100-10140 |
| **set loglevel** | Optional. Sets the logging level. 0 means no logging, while 5 is the maximum log level (the default). e.g. set loglevel=4 |
| **set logfile** | Sets the name or full path of the logfile to direct logging to, e.g. set logfile=C:\temp\ftp.log |
| **echo** | Echos the value of a variable to standard output, whether user-defined or a system variable such as 'protocol'. The variable must be preceded by a '$' to identify it as a variable, e.g. echo $protocol.<br><br>A synonym for echo is *print*. |
| **open [host[:port]]** | Opens the connection to the remote server. The user and password must already be set via the set command. If the remote host is not already set, it must be supplied here (and will override any previous setting for this connection attempt). The port number can also be optionally supplied (but the host must be provided). e.g. `open edtmobile:21`<br><br>`Can also use connect.` |
| **binary** | Change the current transfer mode to binary mode. Binary is the default.<br><br>Can use *bin* as an abbreviation. |
| **ascii** | Change the current transfer mode to ASCII mode. Binary is the default.<br><br>Can use *asc* as an abbreviation. |
| **pwd** | Print the current remote working directory. |
| **dir [remotedir]** | Perform a detailed listing of the current directory if no directory is supplied as an argument, or of the supplied remote directory.<br><br>See *ls* for a similar command. A synomyn of dir is *ls -l*. |
| **ls [remotedir]** | Perform a simple listing (just filenames) of the current directory if no directory is supplied as an argument, or of the supplied remote directory. If *ls -l i*s used, a detailed listing is supplied.<br><br>See *dir* for a similar command. A synomyn of *ls -l is dir*. |
| **cd remotedir** | Change the current remote working directory to *remotedir*. |
| **cdup** | Change the current remote working directory to its parent directory. |
| **mkdir remotedir** | Create the remotedir directory. |
| **rmdir remotedir** | Delete the remotedir directory. It must be empty. |
| **get remotefile [localfile]** | Download a remotefile to the local host. The localfile parameter can be a filename or the full path of a local file. If it is a filename, the current local working directory is prepended.<br><br>Note that the localfile parameter is optional - if not supplied, the downloaded file is saved in the current local working directory with the name of remotefile.<br><br>The remotefile parameter can be a filename or a path. Not all servers support the use of a path - in this case navigate to the correct remote directory using *cd*. |

| | |
|---|---|
| **put localfile [remotefile]** | Upload a local file to the remote host. The localfile parameter can be a filename or the full path of a local file. If it is a filename, the current local working directory is prepended.<br><br>The remotefile parameter should be a filename or a path. Not all servers support the use of a path - in this case navigate to the correct remote directory using cd.<br><br>If the remotefile parameter is not supplied, the local file name will be used. |
| **append localfile [remotefile]** | Append the contents of a local file to a remote file (if it exists) on the remote host. The localfile parameter can be a filename or the full path of a local file. If it is a filename, the current local working directory is prepended.<br><br>The remotefile parameter should be a filename or a path. Not all servers support the use of a path - in this case navigate to the correct remote directory using cd.<br><br>If the remotefile parameter is not supplied, the local file name will be used. |
| **rename remotefile1 remotefile2** | Rename a remote file from remotefile1 to remotefile2. Note that both parameters should be filenames, not full paths.<br><br>Can use *ren* as an abbreviation. |
| **lrename localfile1 localfile2** | Rename a local file from localfile1 to localfile2. Note that paths can be used as well as filenames.<br><br>Can use *lren* as an abbreviation. |
| **delete remotefile** | Delete a remote file in the current remote working directory.<br><br>Can use *del* as an abbreviation. |
| **ldelete localfile** | Delete a local file in the current local working directory, or specify a path for the file to be deleted.<br><br>Can use *ldel* as an abbreviation. |
| **mdelete *wildcard*** | Delete multiple remote files that match the wildcard in the current remote working directory.<br><br>Wildcards supported are '?' for a single matching character, and '*' for multiple matching characters. e.g. to delete all text files in the current remote directory, use mdel *.txt<br><br>Can use *mdel* as an abbreviation. |
| **mput *wildcard*** | Upload multiple local files that match the wildcard in the current local working directory to the current remote working directory.<br><br>Wildcards supported are '?' for a single matching character, and '*' for multiple matching characters. e.g. to upload all text files from the current local working directory, use mput *.txt |
| **mget *wildcard*** | Download multiple local files that match the wildcard in the current remote working directory to the current local working directory.<br><br>Wildcards supported are '?' for a single matching character, and '*' for multiple matching characters. e.g. to download all text files from the current remote working directory use mget *.txt |
| **close** | Closes the connection to the remote server.<br><br>Can also use bye, quit, exit or disconnect. |
| **quote *command*** | Sends the quoted command to the server and displays the response. |
| **site *parameters*** | Sends a SITE command to the FTP server with the supplied parameters. These parameters can vary widely between FTP servers. This command is not applicable to SFTP servers.<br><br>For example site CHMOD 0600 /home/user/privatefile might be used to change the permissions of a file on an FTP server that supports this feature. |
| **auth** | Change from unencrypted 'plain' FTP into secure FTP. |
| **protect** *clear*<br><br>**or**<br><br>**protect** *private* | Change the protection level of the data channel in FTPS.<br><br>Supported options are clear (or c) to set data channels to unencrypted, and private (or p) to set data channels to encrypted. e.g. protect clear<br><br>Alternatively private has the same effect as protect private, and clear has the same effect as protect clear.<br><br>An abbreviation for protect is prot, which combined with the argument abbreviations means that to set the data channels to unencrypted could be done with prot c, and to set to encrypted could be done with prot p. |
| **clear** | In FTPS, change the protection level of the data channel to private. Identical to protect private. |
| **private** | In FTPS, change the protection level of the data channel to private. Identical to protect private. |

| | |
|---|---|
| **ccc** | In FTPS, clears the control channel, setting it back to plain unencrypted text. This can be useful when using firewalls, which need to be able to inspect the control channel to open data channel ports.<br><br>Once ccc has been called, no more protect commands can be used. This is a security measure inherent in the protocol. |
| **record** *recordfile* | Record subsequent commands to the supplied filename. Most useful from within the shell to record a session. |
| **stop** | Stop recording commands to file. |

\

## Notification macros

Email notifications and process triggers support a variety of macros. Email notifications can use these macros in the *Subject* and *Message* fields, while process triggers can use them in scripts or program arguments. The macros are described in the table below.

| Macro | Description |
|---|---|
| %FilePath% | Full path of file in the virtual file-system |
| %FileName% | Name of file |
| %FolderPath% | Path to folder in the virtual file-syste, |
| %WindowsPath% | Path of file in the Windows file-system |
| %WindowsFolder% | Path to folder in the Windows file-system |
| %WindowsPath% | Full path of file in Windows |
| %OwnerEmail% | Email address of owner of folder |
| %OwnerUserName% | Username of owner of folder |
| %OwnerFullName% | Full name of owner of folder |
| %LoginEmail% | Email address of logged-in user |
| %LoginUserName% | Username of logged-in user |
| %LoginFullName% | Full name of logged-in user |
| %Event% | Event that caused the notification |
| %Notification% | Name of this notification |
| %PathFilter% | File filter of this notification |
| %SiteName% | Name of site |
| %ClientIP% | IP address of client |
| %TransferStatus% | Indicates whether or not an operation succeeded. If it succeeded the value will be 'success', otherwise it will be 'failure'. |
| %Time% | Time of event |
| %Date% | Date of event |
| %DateAndTime% | Date/time of event |
| %Timestamp:format% | Formatted timestamp of event. The word 'format' should be replaced by a .NET date/time formatting string (see here and here ).<br>For example, the macro `%Timestamp:yyyy/MM/dd HH:mm:ss%` would produce a string like "2012-02-08 16:33:21". |

# How to write your own events (Enterprise Edition only)

CompleteFTP supports email and process-trigger event handlers. Event extension classes can be written so that custom events are invoked when particular events occur.

For example, it may be desirable to implement a custom auditing system that logs events to a proprietary database.

See event extensions for more details.

# Web Applications in CompleteFTP

CompleteFTP is not just a fileserver! It supports a rich environment for developing web-applications in Javascript, known as JSS (server-side Javascript). This makes it a powerful platform for developing file-oriented web applications. For example, CompleteFTP could be used for:

1. A photo-printing business that requires users to upload large images. Users would be able to upload photos using any protocol (FTP, SFTP, HTTP); see the photos show up on a web-page; and then organize and caption them interactively.
2. An engineering firm with a mobile workforce where users can upload and download large files via FTP/SFTP, and then organize, annotate and share them via a web-app.
3. A TV station or news website with roving reporters and camera-operators. Again, the video files are very large and are best uploaded via FTP or SFTP, but once uploaded it's useful to be able to organize and annotate them via a web-app.

In all these cases there is a significant advantage in having the same server handle both web-applications and file transfers. This contrasts with existing systems where the web-server and file-server are separate (e.g. Apache/Tomcat) thus requiring extra work for integration/interfacing (e.g. user-authentication, file-systems and inter-process messaging).

CompleteFTP's web-app feature can also be used to develop custom system administration interfaces. Administrators could, for example, in a few lines of HTML and Javascript create a form that allows them to add a CompleteFTP user and set up a directory structure under their home-directory.

To learn how to write JSS scripts, refer to the Introduction to JSS. Features are explained in JSS Features, while details on the API available to JSS developers may be found in the JSS API Guide. Remote procedure calls are central to JSS and are described in detail in JSS Remote Procedure Calls (RPC).

# Introduction to JSS (Javascript Server-Side)

CompleteFTP supports Javascript Server-Side (JSS). This means Javascript (normally only served up for execution in the browser) can be executed on the server, and the results served to the browser.

This opens up server-side development to Javascript developers who previously were confined to working in the browser. It helps leverage their skills much further. JSS is ECMAScript 3 compliant, so nearly all Javascript syntax in common use is supported.

There are three main aspects to JSS:

1. All web application code is in the one language - Javascript. All server-side code is in Javascript.
2. It is very easy for clients (i.e. browser applications) to call Javascript functions on the server. This is known as Remote Procedure Calls (RPC), because the browser makes calls to Javascript functions that are running remotely on the server.
3. It supports serving of basic HTTP requests on the server side to write out HTML pages for display in the browser.

**How does it work?**

When JSS is enabled (see below), any file on the server that has the extension `.jss` will be assumed to contain Javascript and will be interpreted on the server. That's it!

The main features of JSS are described in detail here.

**Why use JSS?**

There are some good reasons for doing so:

1. Javascript developers can now write server-side web applications in the language they are familiar with.
2. JSS is standards compliant, so developing in JSS does not irrevocably commit developers to CompleteFTP.
3. JSS is easy to use!

**Enabling JSS**

A file with a `.jss` extension will be run as JSS if:

1. JSS is enabled on the site serving up the page
2. JSS is enabled for the user who owns the folder that contains the file

If both of these conditions are not satisfied then the file will be served up the same as an non-JSS file.

# JSS Features

This article will summarize the main features of JSS:


1. Serving a basic request
2. Handling user input
3. Building responses using templates
4. Calling a server-side function via RPC
5. Combining simple requests and RPC
6. Asynchronous RPC
7. Uploading and downloading files
8. Including other JSS files
9. Controlling accessibility of JSS functions
10. Extending JSS using .NET
11. Exploring Further


For information about the API available to JSS developers please refer to the JSS API Guide.

## 1. Serving a basic request

A basic HTTP request may handled simply by calling the `write()` method of the `response` object, as follows:

```
response.write("<body><html>Hello world</html></body>");
```

which will respond with an HTML page displaying:

```
Hello world
```

JSS is ECMAScript 3 compliant, so nearly all Javascript syntax in common use is supported, for example:

```
function _getMessage(i) {
 return "Hello world " + i;
}

var html = "<html><body>";
for (var i=0; i<3; i++)
 html += _getMessage(i) + "<br>";
html += "</body></html>";

response.write(html);
```

Here a function is called when building the HTML response. The underscore is prepended to the function to indicate that it's not to be called via RPC. This script will output:

```
Hello world 0
Hello world 1
Hello world 2
```

## 2. Handling user input

User-submitted form data and URL arguments may be accessed via the `request` variable. For example, if a form with a text-box called `firstName` is submitted, then the following may be used to access the value of this field:

```
var firstName = request.form.firstName;
response.write("<body><html>Hello " + firstName + "</html></body>");
```

which, if the value of the `firstName` field is `Jack`, will return:

```
Hello Jack
```

## 3. Building responses using templates

JSS supports templates, which facilitates the separation of Javascript and HTML code into separate files.

For example, if a template file called `hello1.html` contains the following:

```
<html>
<body>
  <form method=post>
   Name: <input type=text name=firstName value={{firstName}}>
   <input type=submit>
  </form>
  <br>
  {{message}}
</body>
</html>
```

and a JSS file called `hello1.jss` contains:

```
var data = {};
```

```
if (request.form.firstName != undefined) {
 data.firstName = request.form.firstName;
 data.message = "Hello " + request.form.firstName;
} else {
 data.firstName = "";
 data.message = "Please enter your name";
}
response.writeUsingTemplateFile("hello1.html", data);
```

then a request to `hello1.jss` will return:

```
Name:  Submit
Please enter your name
```

Notice that the tag `{{message}}` in `hello1.html` was substituted by the field with the same name, `message`, of the Javascript object that was passed to the `writeUsingTemplateFile(file, data)` method.

Once the field has been filled and the button clicked, the following will be returned:

```
Name: world  Submit
Hello world
```

If a template is to be provided as a string rather than a file then the `writeUsingTemplateString(template, data)` may be used.

## 4. Calling a server-side function via RPC

Clients may invoke server-side Javascript functions simply by calling them (or, to be precise, their automatically generated proxies). For example, say the server has a JSS file called `hello2.jss` with the following contents:

```
function hello(name) {
 return "Hello " + name + " from the server";
}
```

This is Javascript code that will be run on the server.

Now, say we have an HTML file called `hello2.html` with the following contents:

```
<html>
<body>
 <script src="hello2.jss"></script>
 <script>
  var message = hello("world");
  document.write(message);
 </script>
</body>
</html>
```

When the HTML is rendered in the browser, the result will be:

```
Hello world from the server
```

This message is **generated on the server by the Javascript function** and returned by the call to the `hello()` function. Notice that all code, client-side and server-side, is Javascript and that no AJAX code was written. This is possible because the request to `hello2.jss` returns the Javascript code required to remotely invoke the `hello()` function in `hello2.jss`.

The Javascript code that's returned by the request to `hello2.jss` is compressed for efficiency. The uncompressed code may be viewed by a request to `hello2.jss?proxy=uncompressed`. It contains the auto-generated proxy-code that the browser needs to call the server function. Note that the JSON-RPC protocol is used (see JSS Remote Procedure Calls (RPC)), so there's no problem modifying the auto-generated proxy-code, or even completely replacing it with, for example, jQuery Ajax code.

Proxies that wrap the functions in a class may be generated by adding a `class` argument. For example, `hello2.jss?class=Hello` will wrap the function in a class called `Hello` such that the hello method may be called as follows:

```
var helloObject = new Hello();
helloObject.hello();
```

Proxies that use promises may be generated by adding a `proxy` argument. For example, `hello2.jss?proxy` will allow methods to be called as follows:

```
hello().then(
    function(message) {
        document.write(message);
    },
    function(err) {
        document.write(err.message);
```

```
    }
);
```

## 5. Combining simple requests and RPC

If a JSS file that contains RPC functions is also to handle basic HTTP requests then the request-handling code should be placed inside a function called `processRequest()`. This function's job is to write the HTTP response that will be returned to the client. For example, the following JSS script, `hello3.jss`, returns the HTML to the client, which then invokes a function by RPC.

```
function hello(name) {
 return "Hello " + name + " from the server";
}

function processRequest() {
 var html = "<html><body>\n";
 html += "<script src="hello3.jss?proxy></script>\n";
 html += "<script>\n";
 html += " var message = hello('world');\n";
 html += " document.write(message);\n";
 html += "</script>\n";
 html += "</body></html>";
 response.write(html);
}
```

Note that the Javascript code that's inside the quotes in the `processRequest()` function will be run on the client, and will call the code that's in the `hello()` function, which will be run on the server.

It is important to note that this server-side code is called in two different contexts. When `hello3.jss` is first called by the browser (without parameters), the default `processRequest()` is called to write the response. But the script referred to in the HTML generated by `processRequest()` is `hello3.jss?proxy`. The "proxy" parameter indicates that rather than calling the default `processRequest()`, the caller wants the RPC proxies generated and returned to the browser so that the `hello` function can be called via RPC.

## 6. Asynchronous RPC

Both synchronous and asynchronous remote procedure calls are supported. If an additional argument is provided then it's assumed that the function is to be called asynchronously and that the additional argument is a callback function. For example, in the above example, the `hello()` function would be called asynchronously like this:

```
function onHello(message) {
 document.write(message);
}

hello("world", onHello);
```

If a second additional argument is provided, it will be assume to be an error-callback function, which will be called if an error occurs. For example,

```
function onHelloSuccess(message) {
 document.write(message);
}

function onHelloError(error) {
 document.write("Error: " + error);
}

hello("world", onHelloSuccess, onHelloError);
```

If no error-callback is provided then a default error-handling function will be used.

Although JSON-RPC is used by default, calls may optionally be made using form-based RPC (see JSS Remote Procedure Calls (RPC)). This may be done by including a flag after the error-handler, e.g.

```
hello("world", onHelloSuccess, onHelloError, false);
```

This tells the proxy-function to use forms instead of JSON-RPC.

## 7. Uploading and downloading files

JSS makes it easy to upload and download files. If the desired response to a request is to return a file from the virtual file-system then this may be achieved by setting the `response.downloadFile` property to the path of the file to be downloaded.

If a request contains one or more files to be uploaded (usually via the `input type=file` element in a form) then CompleteFTP will look for a Javascript function called `getUploadPath(fileName, contentType)` and call it to find out where (in the virtual file-system) the file is to be uploaded. If no such function is found then the file will be uploaded to the same directory as the script, if possible. If the name of this function is not suitable, an alternative function may be specified by preceding the `input type=file` element by another element whose name is `getUploadPath` and whose value is the name of the JSS function to be called.

If an error occurs during upload then a function with the name `errorHandler(errorMessage, errorType)` will be called, if available. Again, an alternative name may be provided by including an element with the name `errorHandler`

and with the name of the desired error-handling function as its value.

## 8. Including other JSS files

Other JSS files may be included via the `include(scriptPath)` function. This function evaluates the contents of the included JSS file in the global scope, regardless of what scope it's call from, so any functions and variables declared in the file will have global scope.

## 9. Controlling accessibility of RPC functions

When developing a complex JSS application it's important to manage the accessibility of the API. All functions in the global namespace will be available via RPC. If this is not desired then the `publish` function may be used. For example, if in our hello example a second function was needed but that function wasn't itself to be accessible via RPC, then the following code would be used:

```
(function() {
    function getSource() {
        return "server";
    }

    function hello(name) {
        return "Hello " + name + " from the " + getSource();
    }

 publish("hello", hello);
})();
```

Since all functions are declared inside a function-block there are no functions in the global namespace until the call to the `publish` function pushes the `hello` function into it. Note that the name of the function and the function itself must be provided separately.

Public APIs are vulnerable to denial of service attacks so by default CompleteFTP restricts the number of invocations to 100 per minute. This may be changed by providing an additional argument to the `publish` function, as follows:

```
publish("hello", hello, { maxCallsPerMinute: 10 });
```

Since one JSS file may be included in another JSS file using the `include` function, it's sometimes desirable to allow another JSS file to invoke a function without also exposing it publically. This may be done by providing a `serverOnly` argument, as follows:

```
publish("getSource", getSource, { serverOnly: true });
```

Functions that are published as `serverOnly` will not be included in proxies and may only be invoked by other JSS scripts.

## 10. Extending JSS using .NET

JSS can call .NET methods that are defined in a JSS extensions - see here.

## 11. Exploring Further

Details on the API available in JSS are in JSS API Guide. Information on the RPC protocol used in JSS may be found in JSS Remote Procedure Calls (RPC).

# JSS Templates

JSS supports templates, which facilitates the separation of Javascript and HTML code into separate files.

JSS templates work by expanding tags in a template using values provided by a hash or object. There are no if statements or any other logic - just tags, which are shown by double curly braces, e.g. {{name}}.

This tag expansion happens on the server-side - in the server-side JSS file, Javascript objects are supplied to a render method that takes the HTML file containing the tags and produces the expanded HTML file. See this example.

**Variables**

Variables are the most basic tag. A {{name}} tag in a basic template will try to find the name key in the current context. If there is no name key, an empty string is substituted.

All variables are HTML escaped by default. If you want to return unescaped HTML, use {{name}}}. You can also use & to unescape a variable, e.g. {{& name}}.

**Sections**

Sections render blocks of text one or more times, depending on the value of the key in the current context. They begin with a pound and end with a slash., e.g. {{#item}} and {{/item}}.

If the key has a value of false or is an empty list, the HTML between the pound and slash is not returned. If the key exists and is non-false, the HTML between the pound and the slash is rendered one or more times - once for every item in the list.

JSS templates are based on Mustache templates. More detailed examples and extended capabilities such as lambdas are explained here.

## JSS Examples

Live JSS examples are hosted on our public server and can be found here.

If you are reading this documentation from a machine that is hosting a running instance of CompleteFTP, you may be able to view the JSS examples on your own server, here.

# JSS API Guide

This document provides an overview of the API available within JSS. Full details are provided in the JSS API Reference.

**1. HTTP Request and Response**

Access to information about an incoming request is through a global variable called request. It provides information on query-arguments, form-arguments and cookies.

Control of the response is provided through a global variable called response. This variable is usually not required in RPCs where the result may simply be returned from the function. When it is required it is usually to write a response using the write method. Other actions may be to return the contents of a file, which may be done using the downloadFile property, which should be set to the path of the file that is to be returned. If forceDownload if set then the browser will offer the user to save the file rather than attempting to render it. Requests may be redirected to other URLs by setting the redirectUrl property.

**2. System**

The global variable, system, represents the CompleteFTP system. It provides access to the file-system, authentication and users, databases and information about the current server and site.

**3. File-System Access**

CompleteFTP's virtual file-system is available via the system.getFile(path) method. This method returns objects of the File class. The fact that an object is returned does not imply that a file exists on that path; this may be checked using the exists() method. Refer to the API reference for more details.

**4. Authentication and Users**

A script may log users into CompleteFTP using the system.login(userName, password) method. This method does not return anything, but will throw an exception if authentication fails. Once a user has logged in, information about them may be access via the system.user property.

**5. Server and Site Information**

Information about the current server is available via system.server and about the current site via system.site

**6. Database Access**

CompleteFTP implements the synchronous part of the Web SQL Database API. Databases may be opened or created using the system.openDatabaseSync(connectionString, version) method. Currently, SQLite, SQL Server Compact, SQL Server, ODBC and OLE DB databases are supported. The first two are embedded databases, which are contained in a single file, so the connection-string to these contain the path to the database-file in CompleteFTP's virtual file-system. For details on connection-strings, refer to the API Reference.

***Example: A contact database***

An SQLite database containing contacts is created as follows:

```
var db = system.openDatabaseSync("~/contacts.sqlite", "");
db.transaction(function(tx) {
 tx.executeSql("CREATE TABLE Contact (ContactID integer PRIMARY KEY, Name text, Phone text);");
});
```

A row is inserted as follows:

```
db.transaction(function(tx) {
 tx.executeSql("INSERT INTO Contact(Name, Phone) VALUES (?, ?)", ['Joe', '123']);
});
```

The code below writes the names and phone numbers of all contacts to the system log:

```
db.readTransaction(function(tx) {
 var res = tx.executeSql("SELECT * FROM Contact ORDER BY Name;");
 for (var i=0; i<res.rows.length; i++) {
     var row = res.rows[i];
  console.log(row.Name + ", " + row.Phone);
 }
});
```

**7. Templates**

The templating engine makes it easy to separate HTML code from Javascript code. The response global variable provides useful templating methods, which are mirrored in the templater global variable (necessary if response can't be used).

**8. Logging**

Text may be appended to the CompleteFTP system log using the console.log(message) method. It's logged at the INFO level.

# JSS RPC Reference

CompleteFTP uses Remote Procedure Calls (RPC) to invoke server-side Javascript from the browser. It does this by automatically generating client-side proxies for calling server-side functions, meaning that adding a remotely invocable server-side function is as easy as adding a client-side function.

Two forms of RPC are supported: JSON-RPC and form-based RPC.

**JSON-RPC**

CompleteFTP uses JSON-RPC 2.0 for client-server communications. JSON-RPC is a minimalist JSON-based RPC protocol. For example, the call to the `hello()` function in the Introduction to JSS is marshalled as

```
{"jsonrpc": "2.0", "method": "hello", "params": ["world"], "id": 1}
```

which returns the result in the following form:

```
{"jsonrpc": "2.0", "result": "Hello world from the server", "id": 1}
```

Errors are returned as:

```
{"jsonrpc": "2.0", "error": {"code": 0, "message": "An error occurred"}, "id": 1}
```

**Form-based RPC**

CompleteFTP also supports form-based RPC via standard HTTP GET and POST. If an element named `method` is included in the form then the Javascript function with that name will be called. The arguments in the call will match the elements in the form - in the same order. Thus the `hello()` function in the Introduction to JSS may be called by HTTP-POST using the following HTML:

```
<html>
<body>
  <form method=post>
   <input type=hidden name=method value=hello>
   <input type=text name=firstName>
   <input type=submit>
  </form>
</body>
</html>
```

Alternatively it may be invoked by HTTP-GET via the following URL:

```
http://hostname/directory/script.jss?method=hello&firstName=world
```

In both these cases the result is returned in the JSON-RPC format, i.e.

```
{"jsonrpc": "2.0", "result": "Hello world from the server", "id": 1}
```

## Custom extensions
**(requires Enterprise Edition)**

With CompleteFTP custom extensions you can customize the behaviour of CompleteFTP by integrating your own .NET code. Developing an extension involves creating a DLL and registering it with CompleteFTP Manager (see here).

### Types of Extensions

Five types of extensions are supported:

**Authentication extensions**
Use this type of extension to implement any user-name/password authentication scheme. The server calls methods in your class to find out whether or not a particular user-name/password combination is valid. Your class can do whatever it needs to do to work out whether or not it is valid, such as calling a webservice or invoking an RPC.

**File-system extensions**
Replace the regular file-system with one of your own design. Once your file-system extension is associated with a virtual folder, it will be called by the server whenever a client performs file operations (e.g. reading, writing, directory-listing) in that folder. Your class has complete control over how the operations are performed so it could, for example, tap directly into your company database without accessing any real files.

**Custom command extensions**
Add your own commands simply by adding methods to your custom command extension. These commands will be available to clients via SITE commands in FTP/FTPS, via the shell in SSH and via the SSH_FXP_EXTENDED command in SFTP.

**Event extensions**
CompleteFTP comes with e-mail and process-trigger event-handlers. With Event Extensions you can invoke methods on your own .NET classes when particular events occur.

**Custom command scripting**
User-defined commands can also be directly implemented in Javascript. Like custom command extensions, they can be accessed via FTP/FTPS and SSH, but they may also be invoked via HTTP/HTTPS.

### Support

Email-based support is available to developers of extensions. Please direct questions to Enterprise support. In particular, if you find that data and operations that your extensions require are not available from the classes described here then don't hesitate to request them to be made available.

### Development

We can write extensions for you! Please submit your requirements to sales@enterprisedt.com for a quote.

## How to write an extension

A CompleteFTP Extension is a .NET assembly, usually a DLL, that contains a class that extends the one of the CompleteFTP extension base-classes (see specific extension topic). The assembly can be developed in C#, VB.NET or any other .NET language. If you don't already have Visual Studio then you can use one of Microsoft's free Visual Studio Express products (e.g. Visual C# or Visual Basic).

Note that we can write extensions for you! Please email your requirements to sales@enterprisedt.com for a quote.

## Creating an extension

To create a CompleteFTP extension:

1. Open Visual Studio, Visual C# Express or Visual Basic Express.
2. Create a new project of type Class Library.
3. Add a reference to the CompleteFTPServer.dll. You can find this in the directory C:\Program Files\CompleteFTP\Server. If you are on a 64-bit machine then it will be in C:\Program Files (x86)\CompleteFTP\Server. Make sure the `Specific Version` property is set to `false`.
4. Added references (`using` or `import`) to the namespaces required for your type of extension.
5. Make sure your new class extends the appropriate base-class.
6. Override the appropriate methods.
7. Add any logging you require.
8. Build the project.

## Registering an Extension in CompleteFTP

Extensions must be registered with CompleteFTP. To do this:

1. Open CompleteFTP Manager
2. Select the Extensions panel
3. Click 'Add extension'
4. In the new row enter the Name of the extension; select the Extension Type; enter the full path to the DLL in the Assembly Path field or leave it blank if it's in the *global Extensions directory* (see below); enter the name of your adapter class, including its namespace; and (optionally) the name of the editor class, including the namespace. Note that the assembly is read by the System user, so this user must have access to the file.
5. Some extension types require you to perform an additional registration step, such as adding a new folder-type, or granting permissions to site-commands.
6. Click the Apply button
7. Test your authenticator by connecting from a client.

Note that if assemblies are placed in the *global Extensions directory* then CompleteFTP will look for them there and the Assembly Path field need not be supplied. The Extensions directory may be found at:

```
%PROGRAMDATA%\Enterprise Distributed Technologies\Complete FTP\Extensions
```

## Reloading an Extension in CompleteFTP

Extensions cannot be automatically reloaded if they are altered. If you need to make a change to the code of the extension then you will need to restart the CompleteFTP service to force it to reload the DLL.

For details on the type of extension that you are developing please go to:

- Authentication extensions
- File-system extensions
- Site-command extensions
- Event extensions
- JSS extensions

# Authentication Extensions

Use this type of extension to implement any user-name/password authentication scheme, or to support public key authentication in SFTP. Authentication extensions are known as *authenticators*. The server calls methods in your class to find out whether or not a particular user-name/password combination or user/public key combination is valid. Your class can do whatever it needs to do to work out whether or not it is valid, such as calling a webservice or invoking an RPC.

## Creating an authentication extension

General instructions on building CompleteFTP extensions may be found here.

Authentication extension classes must extend `EnterpriseDT.Net.FtpServer.Core.Authenticator`.

CompleteFTP includes a number of predefined authenticators, such as the external database authenticator, the gateway authenticator (Enterprise Edition only) and the automatic Windows users (AWU) authenticator.

Note that the order of precedence for authentication is as follows:

1. explicitly defined users
2. external database users, gateway users, and custom authentication extensions
3. automatic Windows users (AWU)

So if a user is explicitly defined in CompleteFTP and also in an extension, then the extension will never be used to authenticate the user.

When an authentication extension is used, the *defaultExtension* user is used as a template for user details once login has been successful.

There are two ways to implement an authentication extension. The simple authentication extension is the most straightforward, and should be attempted first. If more flexibility is required, consider writing an advanced authentication extension.

Note that once an authenticator has been written, it must be registered in CompleteFTP with an Extension Type of authenticator. This is done via the CompleteFTP manager. It must then be enabled by selecting the Enabled check box in the *General User Settings* dialog box, accessible from the Users tab in the manager.

## Simple Authentication Extensions

In many (or even most) cases, authenticators can be written as described below. If this is not flexible enough for a particular application, an advanced authentication extension may be required.

The only method that needs to be implemented is `LoadedUserInfo LoadUserInfo(IUserInfo suppliedUserInfo)`. When the server calls this method in your authenticator, it supplies user information that enables your extension to look up that user's authentication details from an external source. You then return a `LoadedUserInfo` object from this method, which contains the user's home directory, password hash and/or public keys. The authenticator caches this loaded user object, and uses it to authenticate the user.

See the class reference for more details of the classes and interfaces involved.

Example 1 shows sample source code for implementing password authentication. See Example 2 for an example of public key authentication and password authentication.

Optionally, the `void SetPassword(IUserInfo suppliedUserInfo, string oldPassword, string newPassword)` can also be implemented. This can be used to change the user's password in the external user data source. Ultimately, this method is called via the FTP SITE command, CPWD, or via the SSH/SFTP password changing mechanism.

SSH/SFTP also permits servers to force a password change on clients. This can be implemented by setting the `LoadedUserInfo.MustChangePassword` property on the `LoadedUserInfo` object that is returned from `LoadUserInfo`. FTP does not permit the server to force a user to change their password.

See Example 3 for how to force and implement password changes in SSH.

**Example 1**

Below is the source code for a sample authenticator which supports password authentication. For simplicity, the password is hardcoded, whereas in production code, it would be retrieved from a data source. The password is not stored - instead, the PasswordHash field is populated when the Password field is set. The PasswordHash field can be set directly - normally only password hashes would be stored.

```
using EnterpriseDT.Net.FtpServer.Core;

namespace EnterpriseDT.Net.FtpServer.Extensions.Test
{
    public class SimplePasswordAuth : Authenticator
    {
        public override LoadedUserInfo LoadUserInfo(IUserInfo suppliedUserInfo)
        {
            if (suppliedUserInfo.UserName=="fred")
            {
                LoadedUserInfo info = new LoadedUserInfo();
                info.Password = "fredspassword";
                return info;
            }
            else
                return null;
        }
    }
}
```

**Example 2**

Below is the source-code for a sample authenticator which allows a user to log via SFTP using public key authentication or by password authentication. For simplicity, the public keys are hard coded, whereas in production code they would be retrieved from a data source.

```
using EnterpriseDT.Net.FtpServer.Core;

namespace EnterpriseDT.Net.FtpServer.Extensions.Test
{
    public class PublicKeyTest : Authenticator
    {
        public override LoadedUserInfo LoadUserInfo(IUserInfo suppliedUserInfo)
        {
            LoadedUserInfo info = new LoadedUserInfo();
            info.RSAPublicKeys.Add(RSAPublicKey);
            info.DSAPublicKeys.Add(DSAPublicKey);
            info.Password = "mypassword";
            return info;
        }

        private byte[] RSAPublicKey
        {
            get
            {
                string key = "---- BEGIN SSH2 PUBLIC KEY ----\r\n" +
                    "Comment: \"imported-openssh-key\"\r\n" +
                    "AAAAB3NzaC1yc2EAAAABIwAAAIEA44J6LBloMWVvhOjMHZPnmgJWw+UWBl9nFEWa\r\n" +
                    "62IFDrJDg6+kJ2DQD8vOTsQmNjk88O3v+r0r/rr+QrotuLrdjrXBvrRrQNMfEbMo\r\n" +
                    "LhSmUVEFR/Yy3HjVRT6DHhJYPpr1xaXE6++fo5b2ax1zw+d1fPsh53lbAhrCHV9b\r\n" +
```

```
                    "NIOimDk=\r\n" +
                    "---- END SSH2 PUBLIC KEY ----";
                return Encoding.ASCII.GetBytes(key);
            }
        }

        private byte[] DSAPublicKey
        {
            get
            {
                string key = "---- BEGIN SSH2 PUBLIC KEY ----\r\n" +
                    "Comment: \"imported-openssh-key\"\r\n" +
                    "AAAAB3NzaC1kc3MAAACBAPos9tWoXLcd//dOGbaA+1TCO9vEi0jQOQM85j34E4Ua\r\n" +
                    "Sza5yjS3vI9K9XchJirbNYrRQNmgM2yn3fUDdTPU5eES+mZRy9K9qpAesk4Ghpwu\r\n" +
                    "btWc3e0APkQTUAoRHL8yiW1tHrRdV6yrowgKDPrIccnL90wYAZFHmUmwIeiESjTB\r\n" +
                    "AAAAFQDhvm9w83LDeixC3oPW+FOKk673dQAAAIBObehA6t+eRtNTocY1sb7Dly0O\r\n" +
                    "ReeRWo+mHEyUts78ayAN7YFNzTd8UXmUgw8gyGFtO/tXrkeLG46vMhL/0402ek9Z\r\n" +
                    "jNcDq2vF1InYIaOxceuRqg99VGQUqrjEWchIG5egDgtOKRAUtUyK7I52CXG3wN9/\r\n" +
                    "2Oq+WOoUztJCSwgmwwAAAIEAu4G5CHifmoTsBVcObaRkW8UqrTCmz7C84W6AaXA5\r\n" +
                    "uBlwtTIBlAUnKzfqStpC76rucJ6i3R9Nk+gHrDb4v6uA6at2UZDlHZlwPCg88fk7\r\n" +
                    "Nbi5umH9B/QSfm+GQOd+ttD54FOcR+lwmerJ+f1mzSX9v9ZrVi+xJJ6Jp+5NDa7g\r\n" +
                    "KtM=\r\n" +
                    "---- END SSH2 PUBLIC KEY ----";
                return Encoding.ASCII.GetBytes(key);
            }
        }

    }
}
```

**Example 3**

Below is the source-code for a sample authenticator which allows a user to login via SFTP by password authentication, but forces the user to change their password. This is an SSH-specific mechanism that many SSH clients support.

There are three requirements for doing this. Firstly, the `LoadedUserInfo` object must have the `MustChangePassword` property set to true. The `void SetPassword(IUserInfo suppliedUserInfo, string oldPassword, string newPassword)` method must be overridden to allow the new password to be set - throw an `InvalidPasswordException` if the new password does not pass the required checks. And finally, in the *Password policies* dialog found from the link in the *General User Settings* dialog box (accessible from the Users tab in the manager), *Permit password changes* must be selected.

```
using EnterpriseDT.Net.FtpServer.Core;
namespace EnterpriseDT.Net.FtpServer.Extensions.Test
{
    public class ChangePasswordAuth : Authenticator
    {
        private const string PASSWORD = "javaftp";
        private string password = PASSWORD;

        public override LoadedUserInfo LoadUserInfo(IUserInfo suppliedUserInfo)
        {
            LoadedUserInfo info = new LoadedUserInfo();
            info.Password = password;
            info.MustChangePassword = (password == PASSWORD);
            return info;
        }

        public override void SetPassword(IUserInfo suppliedUserInfo, string oldPassword, string newPassword)
        {
            if (oldPassword == newPassword)
                throw new InvalidPasswordException("New password must be different");
            password = newPassword;
        }
    }
}
```

## Advanced Authentication Extensions

Simple authentication extensions can usually be used to accomplish what extension developers require. However at times more complicated scenarios require more flexibility.

As with simple extensions, advanced authentication extension classes must extend `EnterpriseDT.Net.FtpServer.Core.Authenticator`.

However instead of overriding the `LoadUserInfo` method, they override the `CheckUserName` and `Authenticate` methods. In the `EnterpriseDT.Net.FtpServer.Core.Authenticator` base class implementation, `LoadUserInfo` is called by `CheckUserName`. But `LoadUserInfo` need not be overridden - instead `CheckUserName` can be overridden and `LoadUserInfo` need not be called at all.

Similarly, the `Authenticate` method is implemented in the `Authenticator` class, and if it is provided in `CheckUserName` or indirectly via `LoadUserInfo`, the `LoadedUserInfo` class containing the user's data will automatically used to authenticate the user. However `Authenticate` can be overridden to authenticate in whatever way the extension writer prefers.

The primary requirement when overriding `CheckUserName` is to ensure that `IsValidUserName` property of the supplied `IUserInfo` argument is set to true for a valid user. When overriding `Authenticate`, ensure that the `IsCorrectPassword` property of the supplied `IAuthenticationInfo` argument is set to true for password authentication, or `IsValidKey` is set to true for public key authentication. `IAuthenticationInfo.AuthenticationMethod` can be inspected to determine if password or public key authentication is being used. `IsValidRSAKey` and `IsValidDSAKey` can be used to determine if a key blob is the correct public key matching what the user has supplied as part of their public key authentication process.

See the class reference for more details of the classes and interfaces involved.

**Example 1**

Below is the source-code for a sample authenticator which allows a user to log in with the username, `myusername`, and the password, `mypassword`. The user's home directory will be `C:\Temp` if the `defaultExtension`'s home-directory is set to `%ExternalHomeFolder%`.

```
using EnterpriseDT.Net.FtpServer.Core;

namespace MyTestAuthenticator
{
  public class PasswordAuthenticator : Authenticator
  {
    public override void CheckUserName(IUserInfo userInfo)
    {
      if (userInfo.UserName == "myusername")
        userInfo.IsValidUserName = true;
    }

    public override void Authenticate(IAuthenticationInfo authInfo)
    {
      if (authInfo.AuthenticationMethod == AuthenticationMethod.Password &&
          authInfo.Password == "mypassword")
      {
        authInfo.IsCorrectPassword = true;
        authInfo.HomeDirectory = "C:\\Temp";
      }
    }
  }
}
```

**Example 2**

Below is the source-code for a sample authenticator which allows a user to log in with the username, `myusername` via public key authentication. For simplicity, the public keys of the user are hard coded, whereas in production code they would be retrieved from a data source. The user's home directory will be `C:\Temp` if the `defaultExtension`'s home-directory is set to `%ExternalHomeFolder%`.

```
using EnterpriseDT.Net.FtpServer.Core;

namespace MyTestAuthenticator
{
  public class PublicKeyAuthenticator : Authenticator
  {
    public override void CheckUserName(IUserInfo userInfo)
    {
      if (userInfo.UserName == "myusername")
        userInfo.IsValidUserName = true;
    }

    public override void Authenticate(IAuthenticationInfo authInfo)
    {
      if (authInfo.AuthenticationMethod == AuthenticationMethod.PublicKey)
      {
         if (authInfo.KeyAlgorithm == PublicKeyAlgorithm.DSA)
         {
```

```
            authInfo.IsValidKey = IsValidDSAKey(DSAPublicKey, authInfo);
         }
         else
         {
            authInfo.IsValidKey = IsValidRSAKey(RSAPublicKey, authInfo);
         }
         if (authInfo.IsValidKey)
            authInfo.HomeDirectory = "C:\\Temp";
      }
   }

   private byte[] RSAPublicKey
   {
     get
     {
       string key = "---- BEGIN SSH2 PUBLIC KEY ----\r\n" +
           "Comment: \"imported-openssh-key\"\r\n" +
           "AAAAB3NzaC1yc2EAAAABIwAAAIEA44J6LBloMWVvhOjMHZPnmgJWw+UWBl9nFEWa\r\n" +
           "62IFDrJDg6+kJ2DQD8vOTsQmNjk88O3v+r0r/rr+QrotuLrdjrXBvrRrQNMfEbMo\r\n" +
           "LhSmUVEFR/Yy3HjVRT6DHhJYPpr1xaXE6++fo5b2ax1zw+d1fPsh53lbAhrCHV9b\r\n" +
           "NIOimDk=\r\n" +
           "---- END SSH2 PUBLIC KEY ----";
       return Encoding.ASCII.GetBytes(key);
     }
   }

   private byte[] DSAPublicKey
   {
     get
     {
       string key = "---- BEGIN SSH2 PUBLIC KEY ----\r\n" +
           "Comment: \"imported-openssh-key\"\r\n" +
           "AAAAB3NzaC1kc3MAAACBAPos9tWoXLcd//dOGbaA+1TCO9vEi0jQOQM85j34E4Ua\r\n" +
           "Sza5yjS3vI9K9XchJirbNYrRQNmgM2yn3fUDdTPU5eES+mZRy9K9qpAesk4Ghpwu\r\n" +
           "btWc3e0APkQTUAoRHL8yiW1tHrRdV6yrowgKDPrIccnL90wYAZFHmUmwIeiESjTB\r\n" +
           "AAAAFQDhvm9w83LDeixC3oPW+FOKk673dQAAAIBObehA6t+eRtNTocY1sb7Dly0O\r\n" +
           "ReeRWo+mHEyUts78ayAN7YFNzTd8UXmUgw8gyGFtO/tXrkeLG46vMhL/04O2ek9Z\r\n" +
           "jNcDq2vF1InYIaOxceuRqg99VGQUqrjEWchIG5egDgtOKRAUtUyK7I52CXG3wN9/\r\n" +
           "2Oq+WOoUztJCSwgmwwAAAIEAu4G5CHifmoTsBVcObaRkW8UqrTCmz7C84W6AaXA5\r\n" +
           "uBlwtTIBlAUnKzfqStpC76rucJ6i3R9Nk+gHrDb4v6uA6at2UZDlHZlwPCg88fk7\r\n" +
           "Nbi5umH9B/QSfm+GQOd+ttD54FOcR+lwmerJ+f1mzSX9v9ZrVi+xJJ6Jp+5NDa7g\r\n" +
           "KtM=\r\n" +
           "---- END SSH2 PUBLIC KEY ----";
       return Encoding.ASCII.GetBytes(key);
     }
   }
  }
}
```

## File-System Extensions

A file-system extension is a .NET assembly (usually a DLL) that contains a class that extends the class `EnterpriseDT.Net.FtpServer.FileSystem.FileSystemAdapter`. The assembly can be developed in C#, VB.NET or any other .NET language. If you don't already have Visual Studio then you can use one of Microsoft's free Visual Studio Express products (e.g. Visual C# or Visual Basic).

## Creating a file-system extension

General instructions on building CompleteFTP extensions may be found here. In addition to the registration steps described in the general instructions, a file-system extensions must have a folder-type defined. The folder-type determines which extension should be used for a specific folder-type. Once a new folder-type is added, it will be displayed in the list of folder-types that is displayed when a user adds a new folder to the virtual file-system.

Authentication extension classes must extend `EnterpriseDT.Net.FtpServer.FileSystem.FileSystemAdapter` or one of its subclasses, e.g. `WindowsAdapter`, `FolderAdapter` or `SpecialFolderAdapter`. The base-class, `FileSystemAdapter`, is a concrete class but it disallows all operations, some of its methods must therefore be overridden to make it functional. For example, if directory-listing and downloading are to be supported then `GetFileInfos` and `GetReadStream` must be implemented at a mininum. Many FTP clients do things like checking sizes and modification dates before downloading, so methods like `GetFileLength` and `GetModifiedTime` may also need to be implemented.

**Example**

Below is the source-code for a sample file-system extension whose only function is to display file-names in upper case.

```
using EnterpriseDT.Net.FtpServer.Core;
using EnterpriseDT.Net.FtpServer.FileSystem;
using EnterpriseDT.Net.FtpServer.FileSystem.Windows;

namespace MyTestAdapter
{
  public class UpperCaseFileNameAdapter : FolderAdapter
  {
    public override VirtualFileInfo[] GetFileInfos(ISession session,
                                                   IFileSystemNode node,
                                                   string path, string pattern)
    {
      VirtualFileInfo[] files = base.GetFileInfos(session, node, path, pattern);
      foreach (VirtualFileInfo f in files)
        f.Name = f.Name.ToUpper();
      return files;
    }
  }
}
```

## Creating an editor for a file-system extension

A file-system extension can optionally have a visual editor associated with it. This editor simply needs to pop up a form when a particular method is called. Extension editors must extend the class `EnterpriseDT.Net.FtpServer.FileSystem.FileSystemAdapterEditor`. They should be placed in the same assembly as the FileSystemnAdapter extension itself. At a minimum they must override the `IsUserConfigurable` property and the `ShowConfigDialog` method. The `IsUserConfigurable` method should return `true`. The `ShowConfigDialog` takes an argument of type `IFileSystemNode`, which represents a folder in the virtual file-system. The job of the `ShowConfigDialog` method is to change this the `IFileSystemNode`'s `Configuration` property and return true if it has changed it.

Every method of `FileSystemAdapter` takes an argument of the same type, i.e. `IFileSystemNode`. Thus every method will have access to the user-configurable `Configuration` property. This is how, for example, each Windows folder in the virtual file-system is mapped to a specific Windows folder on the hard-disk.

**Example**

The following example opens a `FolderBrowserDialog` and allows the user to select a Windows path. Notice that the `node.Configuration` property is only set if the OK-button was pressed by the user.

```
public class TestAdapterEditor : FileSystemAdapterEditor
{
  public override bool ShowConfigDialog(IWin32Window parent, IFileSystemNode node)
  {
    FolderBrowserDialog browser = new FolderBrowserDialog();
    if (node.Configuration!=null)
      browser.SelectedPath = node.Configuration;
    if (browser.ShowDialog(parent) == DialogResult.OK)
    {
      node.Configuration = browser.SelectedPath;
      return true;
    }
    else
      return false;
  }
}
```

## Overview of File-System Adapter Classes

All file-system adapters must extend directly or indirectly from `FileSystemAdapter`. If you want your adapter to

access the Windows file-system then it's probably much easier to extend from one of its subclasses, such as `FolderAdapter`. For example, if you only want to implement some special mapping between the virtual file-system and the Windows file-system then you can simply override the `GetWindowsPath` method, which is responsible for mapping virtual paths into real paths.

**FileSystemAdapter**

**namespace:** `EnterpriseDT.Net.FtpServer.FileSystem`
**editor:** `None`

Every file or directory operation that the user initiates is done by means of a `FileSystemAdapter`. This includes uploading, downloading, renaming and deleting files, as well as creating, listing and deleting directories. Most of these operations are implemented by a single method. The class itself can be instantiated, but does nothing of value. It yields empty listings and allows no uploads, downloads, deletes, etc. Any methods or all methods may be overridden in subclasses to add functionality. As stated previously, its often desirable to extend `FolderAdapter` since it already implements all methods required for interfacing with the Windows file-system, but may be modified as required.

Most methods of `FileSystemAdapter` take at least the following three arguments: `session`, `node` and `path`. The `session` contains information about the user's current session; the `node` has information about the folder in the virtual file system that the user is accessing; and the `path` is the path relative to the last virtual folder on the path. So if, for example, the user is accessing the file `/a/b/c/d/e.txt`, which is in a subfolder of the virtual directory, `/a/b/c`, then the path argument will be `d/e.txt`. The value of the property `node.Configuration` is the value set by the adapter editor (see above) when the virtual folder was created by the user.

### WindowsAdapter extends FileSystemAdapter (abstract)

**namespace:** `EnterpriseDT.Net.FtpServer.FileSystem.Windows`
**editor:** `None`

Provides all the functionality necessary for CompleteFTP to access the Windows file-system **except** for the mapping of paths in the virtual file-system into real Windows paths. When it needs to map a path in the virtual file-system to a Windows path it calls its own abstract method, `GetWindowsPath`. The main job of the subclasses of `WindowsAdapter` is to implement this method.

### FolderAdapter extends WindowsAdapter

**namespace:** `EnterpriseDT.Net.FtpServer.FileSystem.Windows`
**editor:** `FolderAdapterEditor`

This is the main adapter for the Windows file-system. The editor sets the `node.Configuration property` to an absolute Windows path. The `GetWindowsPath` concatenates this absolute Windows path and the path passed to the `GetWindowsPath` method to return the full Windows path of the folder on in the Windows file-system.

### SpecialFolderAdapter extends WindowsAdapter

**namespace:** `EnterpriseDT.Net.FtpServer.FileSystem.Windows`
**editor:** `SpecialFolderAdapterEditor`

Adapter for special Windows folder, such as the My Documents folder. The editor sets the `node.Configuration` property to a tag indicating which special folder is required. The `GetWindowsPath` returns the real Windows path of this folder for the user whose session is being accessed.

## Overview of Other Classes

**ISession**

**namespace:** `EnterpriseDT.Net.FtpServer.FileSystem`

Provides information about the current user-session, including the name of the user and the IP address of the client.

**IFileSystemNode**

**namespace:** `EnterpriseDT.Net.FtpServer.FileSystem`

A file-system node is a folder in the virtual file-system. Every folder that you see in the virtual file-system view has a single file-system node associated with it. The most important property of `IFileSystemNode` is `Configuration`, which stores the path or other configuration that the visual editor set when the user configured the folder in CompleteFTP Manager.

**VirtualFileInfo**

**namespace:** `EnterpriseDT.Net.FtpServer.FileSystem`

Used to pass directory listing information from the FileSystemAdapter to CompleteFTP.

**VirtualFileMode**

**namespace:** `EnterpriseDT.Net.FtpServer.FileSystem`

## Logging

The class `EnterpriseDT.Util.Debug.Logger` may be used to write messages to the CompleteFTP log. This can be very useful when debugging your adapter. To use this class, create a static instance of it in you adapter class and call its Error, Warn or Debug method to write something to CompleteFTP's log file. For example:

```
using EnterpriseDT.Util.Debug;

public class UpperCaseFileNameAdapter : FolderAdapter
{
 private Logger log = Logger.GetLogger("UpperCaseFileNameAdapter");
```

```
 public override VirtualFileInfo[] GetFileInfos(ISession session, IFileSystemNode node, string path, string pattern)
 {
  log.Debug("Listing requested");
  ...
 }
}
```

## Class Reference

**Class: FileSystemAdapter**

`bool DirectoryExists(ISession session, IFileSystemNode node, string path)`
Returns `true` if a directory exists and `false` otherwise.

`void CreateDirectory(ISession session, IFileSystemNode node, string path)`
Create a directory at the given path.

`VirtualFileInfo[] GetFileInfos(ISession session, IFileSystemNode node, string path, string pattern)`
Return an array of `VirtualFileInfo` objects corresponding to the files in the given directory that match the given pattern.

`void DeleteDirectory(ISession session, IFileSystemNode node, string path)`
Delete the given directory.

`bool FileExists(ISession session, IFileSystemNode node, string path)`
Returns `true` if a file exists and `false` otherwise.

`void DeleteFile(ISession session, IFileSystemNode node, string path)`
Delete the given file.

`bool IsFileLengthKnown(ISession session, IFileSystemNode node, string path)`
Returns `true` if the length of the given file is known and `false` otherwise.

`long GetFileLength(ISession session, IFileSystemNode node, string path)`
Returns the length of the given file in bytes.

`DateTime GetModifiedTime(ISession session, IFileSystemNode node, string path)`
Returns the time when the given file was last modified.

`void SetModifiedTime(ISession session, IFileSystemNode node, string path, DateTime dateTime)`
Set the last-modified time of the given file.

`Stream GetReadStream(ISession session, IFileSystemNode node, string path)`
Returns a stream that may be used for reading the given file (used for downloads).

`void OnReadStreamClosed(ISession session, IFileSystemNode node, string path, Stream stream)`
Called when the read-stream (created by GetReadStream) is closed.

`Stream GetWriteStream(ISession session, IFileSystemNode node, string path, VirtualFileMode mode)`
Returns a stream that may be used for writing to the given file (used for downloads).

`void OnWriteStreamClosed(ISession session, IFileSystemNode node, string path, Stream stream)`
Called when the write-stream (created by GetWriteStream) is closed.

`bool MoveFile(ISession session, IFileSystemNode fromNode, string fromPath, FileSystemAdapter toAdapter, IFileSystemNode toNode, string toPath)`
Move the given file from one location in the virtual file-system (defined by fromNode and fromPath) to another location (defined by toAdapter, toNode and toPath). For simple file renaming, the fromNode will be the same as the toNode and the toAdapter will be the 'this' adapter.

`string GetAdapterFilePath(ISession session, IFileSystemNode node, string path)`

`void Initialize(IPlugInInfo info)`
Called when the extension is first created.

`void Dispose()`
Called when the extension is disposed of.

## Web-app Extensions

Use this type of extension to implement a web application extension. This is a class which is passed an HTTP request, does something with it, and passes back an HTTP response that is sent back to the browser. It is only applicable to the HTTP and HTTPS protocols.

## Creating a web-app extension

General instructions on building CompleteFTP extensions may be found here.

Web-app extension classes must extend `EnterpriseDT.Net.FtpServer.Http.WebAppExtension`.

**Example**

Below is the source code for a sample web-app extension that simply displays a "Hello world" message in a browser.

```
using EnterpriseDT.Net.FtpServer.Http;
using EnterpriseDT.Util.Debug;
using EnterpriseDT.Net.FtpServer.FileSystem;

namespace EnterpriseDT.Demo
{
    public class WebAppTest : WebAppExtension
    {
        private Logger log = Logger.GetLogger("WebAppTest");

        public override void Process(WebAppRequest request, WebAppResponse response)
        {
            log.Info("Process called");
            response.SetResponseText("<html><body><h1>Hello world</h1></body></html>");
            response.ContentType = "text/html";
        }
    }
}
```

## How to deploy a web-app extension

## Custom Command Extensions

A custom command extension is a .NET assembly (usually a DLL) that contains a class that extends the class `EnterpriseDT.Net.FtpServer.Core.CustomCommands`. The assembly can be developed in C#, VB.NET or any other .NET language. If you don't already have Visual Studio then you can use one of Microsoft's free Visual Studio Express products (e.g. Visual C# or Visual Basic).

## Creating a custom command extension

General instructions on building CompleteFTP extensions may be found here. In addition to the registration steps described in the general instructions, permissions must also be granted for users and/or groups to access a custom command extension, otherwise no users will be able to invoke the commands.

Permissions for custom command extensions are managed in the Custom Command Permissions tab of the Extensions panel in CompleteFTP Manager. The commands in a custom command extension may be executed by a user that's been directly given or is a member of a group that has been given permission to do so.

Commands are added to a custom command extension simply by adding methods. By default the name of the method becomes the name of the command. If a `CustomCommandName` attribute is defined then that becomes the name of the command. All names are case-insensitive. The arguments of the method become the arguments of the command. All the arguments must be strings. Custom commands may return one or more lines of text.

The way in which a command is executed by a client depends on the protocol. In FTP custom commands must be preceded by SITE. For example, the inbuilt change password command may be invoked as follows:

```
ftp> site cpwd mypassword
200-CPWD successful
200 SITE command successful.
ftp>
```

Note that if you're using Microsoft ftp.exe then you need to enter `"quote site mypassword"`.

**Example**

The following example implements the ubiquitous Hello World example as a Custom command:

```
using EnterpriseDT.Net.FtpServer.Core;

namespace SampleExtensions
{
    public class HelloWorldCustomCommands : CustomCommands
    {
        public string HelloWorld()
        {
            return "Hello world!";
        }

        [CustomCommandName("HelloWorld2")]
        public string HelloWorldWithArgs(string argument1, string argument2)
        {
            return "Hello world!\n" + argument1 + "\n" + argument2;
        }
    }
}
```

The first method has no arguments and defines no `CustomCommandName` attribute. This means that it must be called by its full name, "HelloWorld" (case is ignored). The second method has two arguments, so two arguments must be entered when the command is executed by the client. This method also defines a command-name, so that name, "HelloWorld2", must be used when the command is executed by the client. In addition, it illustrates that multiple lines may be included in a reply.

## Event Extensions

An event extension is a .NET assembly (usually a DLL) that contains a class that extends the class `EnterpriseDT.Net.FtpServer.Trigger.Trigger`. The assembly can be developed in C#, VB.NET or any other .NET language. If you don't already have Visual Studio then you can use one of Microsoft's free Visual Studio Express products (e.g. Visual C# or Visual Basic).

## Creating an event extension

General instructions on building CompleteFTP extensions may be found here.

Event extension classes must extend `EnterpriseDT.Net.FtpServer.Trigger.Trigger` and implement the `OnTriggerEvent` method. This method is called each time one of the following events occur: `LogIn`, `LogOut`, `ChangeDirectory`, `UploadFile`, `DownloadFile`, `MoveFile`, `DeleteFile`, `CreateFolder`, `DeleteFolder` and `MoveFolder`.

The `OnTriggerEvent` is passed a single argument of type `TriggerEvent`. This class contains various information about the event that occurred and the user-session in which it occurred.

**Example**

The following example implements a minimal auditing event extension, which simply logs all events to a CSV file in the user's temporary directory (i.e. C:\Users\{username}\AppData\Local\Temp):

```
public class AuditingEventHandler : Trigger
{
    public override void OnTriggerEvent(TriggerEvent eventInfo)
    {
        string auditFilePath = Path.Combine(Path.GetTempPath(), "audit.csv");
        using (StreamWriter auditFile = File.AppendText(auditFilePath))
        {
            auditFile.WriteLine("{0:yyyy-MM-dd hh:mm:ss},{1},{2},{3},{4}",
                DateTime.Now, eventInfo.EventType, eventInfo.Session.UserName,
                eventInfo.Path, eventInfo.OldPath);
        }
    }
}
```

## Class Reference

**Class: Trigger**

All event extensions must extend this class. Only the `OnTriggerEvent` method needs to be implemented. It is called every time an event occurs. If any initialization needs to be done then that may be placed in the `Initialize` method, and any required clean-up code may be placed in the `Dispose` method.

`void OnTriggerEvent(TriggerEvent eventInfo)`
Called every time an event occurs.

`void Initialize(IPlugInInfo info)`
Called when the Trigger is created. Perform any initialization tasks here. `info.PlugInConfiguration` contains the configuration entered by the user.

`void Dispose()`
Called when the Trigger is disposed of. Perform any clean-up tasks here

**Class: TriggerEvent**

Contains information describing a particular event

`TriggerEventType EventType`
Type of event. Possible values are `LogIn`, `LogOut`, `ChangeDirectory`, `UploadFile`, `DownloadFile`, `MoveFile`, `DeleteFile`, `CreateFolder`, `DeleteFolder` and `MoveFolder`.

`ISession Session`
Provides information about the current user-session, including the name of the user and the IP address of the client.

`string Path`
Path of file (if any)

`string OldPath`
Previous path of file (valid for `MoveFile` and `MoveFolder` events only).

## JSS Extensions

JSS extensions may be used to call into .NET from JSS thus offering wide extensibility to the JSS platform. It might be used, for example, to wrap a simple .NET function so that it may be used in JSS or as a bridge between JSS and a .NET API for a web-service.

A JSS extension is a .NET assembly (usually a DLL) that contains a class that extends the class `EnterpriseDT.Net.FtpServer.Core.PlugIn` and has one or more methods with the `JavascriptInvokable` attribute. The assembly can be developed in C#, VB.NET or any other .NET language. If you don't already have Visual Studio then you can use one of Microsoft's free Visual Studio Express products (e.g. Visual C# or Visual Basic).

## Creating an event extension

General instructions on building CompleteFTP extensions may be found here.

JSS extension classes must extend `EnterpriseDT.Net.FtpServer.Trigger.PlugIn`. Each method that is to be exposed to JSS must have a `JavascriptInvokable` attribute. If the `JavascriptInvokable` attribute specifies a name then this is the name by which the method must be called from JSS. If no name is specified then the name of the .NET method is used.

Arguments may be passed to a method and values returned. Javascript numbers will be cast to .NET doubles; Javascript string will be case to .NET strings; arrays will be cast to `List<object>`; and Javascript objects will be cast to `Dictionary<string, object>`.

**Example**

The following example implements methods that calculate hashes using two algorithms, MD5 and SHA1. Notice that the `JavascriptInvokable` attributes specify the names that should be used to call them from JSS, i.e. `calculateMD5` and `calculateSHA1`.

```
public class HashFunctions : PlugIn
{
    [JavascriptInvokable("calculateMD5")]
    public string CalcMD5(string s)
    {
        return FormsAuthentication.HashPasswordForStoringInConfigFile(s, "MD5");
    }

    [JavascriptInvokable("calculateSHA1")]
    public string CalcSHA1(string s)
    {
        return FormsAuthentication.HashPasswordForStoringInConfigFile(s, "SHA1");
    }
}
```

## Custom Command Script

User-defined commands can be implemented in Javascript - each command is implemented by a Javascript function. To add commands, go to the Extensions tab of the CompleteFTP manager, and select Custom Command Script. *Add command* can be used to add a new command. /p>

The commands may be accessed as SITE commands in FTP/FTPS and as regular commands in SSH terminal. They may also be invoked via HTTP/HTTPS using a URL of the format shown below and the result is returned in JSON format.

```
http://hostname/CustomCommand/commandName?arg1=val1
```

The following function defines a command called 'Hello' which takes one argument:

```
function Hello(who)
{
    return 'Hello ' + who;
}
```

All arguments are strings.

Commands added in this way are available to all users who have permission for the 'CustomCommandScript' extension. These permissions may be modified in the Permissions tab of the Extensions panel.

.NET objects may be instantiated using the 'new' keyword followed by the fully qualified name, e.g.

```
function Hello(who)
{
    var b = new System.Text.StringBuilder();
    b.Append('Hello').Append(' ').Append(who);
    return b.ToString();
}
```

Static methods and properties of .NET objects may be accessed using their fully qualified name, e.g.

```
function HostName()
{
    return System.Environment.MachineName;
}
```

## How to add logging to an extension

The class `EnterpriseDT.Util.Debug.Logger` may be used to write messages to the CompleteFTP log. This can be very useful when debugging your extension. To use this class, create a static instance of it in your extension class and call its Error, Info, Warn or Debug method to write something to CompleteFTP's log file. For example:

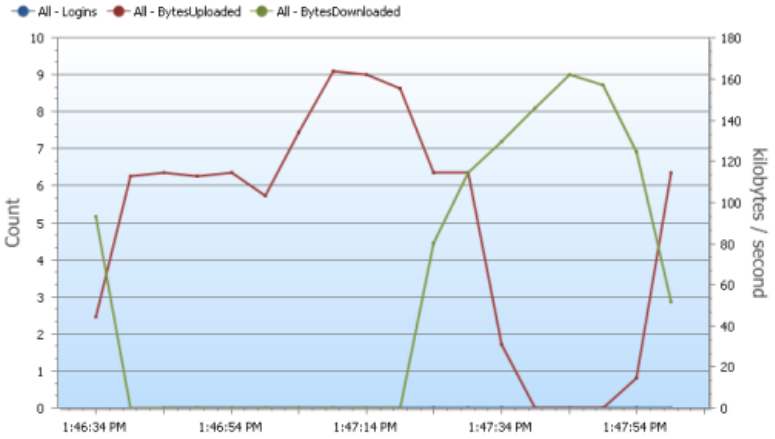
```
using EnterpriseDT.Util.Debug;

public class UpperCaseFileNameAdapter : FolderAdapter
{

  private static Logger log = Logger.GetLogger("UpperCaseFileNameAdapter");

  public override VirtualFileInfo[] GetFileInfos(ISession session, IFileSystemNode node, string path, string pattern)
  {

    log.Debug("Listing requested");

    ...

  }
}
```

# Monitoring

The *Monitoring* panel in the CompleteFTP manager displays performance statistics (in Professional and Enterprise Editions only), details about the currently connected users, the IP addresses that have been auto-banned, recent diagnostic messages, and the server log files. By default the monitoring tabs refresh automatically. This can be disabled.

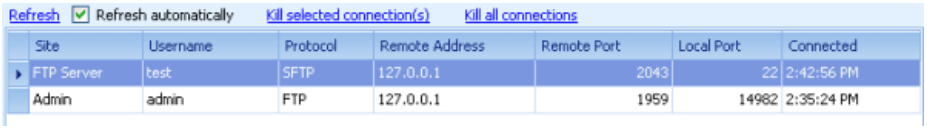
**Performance**

A variety of real-time statistics can be charted, such as the number of connections, the number of file uploads and downloads, and the bytes uploaded and downloaded. These can be charted cumulatively and for each protocol.



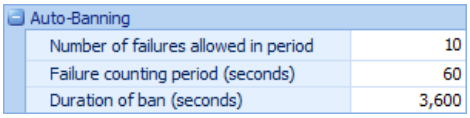

**Connections**

In the *client connections* tab the current connections to the server are shown. The *Kill selected connection* option can be used to immediately disconnect a user from the server. The *admin* user cannot be killed.




**Auto-bans**

In the *auto-banned IP addresses* tab, all currently auto-banned IP addresses are shown. Selecting an IP address and clicking *Remove selected ban(s)* will unban an IP address. Choosing *Permanently ban select IP address(es)* will add a permanent filter banning the selected IP addresses. In the Professional and Enterprise Editions, auto-banning can be configured as shown below and as described here.



# How to view the server log files

CompleteFTP has a sophisticated logging system based on log4net.

Logs are extremely useful for diagnosing problems or simply for monitoring what is happening on the server. Logs can be viewed remotely via the CompleteFTP manager. The CompleteFTP manager also produces its own log file.

The Diagnostics tab in the Monitoring panel displays the server's log files. A log file may be viewed by selecting it in the top window that displays the logs.

The *Diagnostics.log* contains all available logging (depending on the level set). The *Errors.log* contains the last few hundred lines of logging after an error has been logged - it does not contain the full log file.

As the log files are downloaded to the manager (as the manager may be running on a different machine), it is necessary to select the *Refresh* option on the right hand side of the view of the log file to see the latest logging information if some time has passed after initially viewing the file.

The *logging level* can be set in this window. By default, it is set to "Information". If problems occur, this should be set to "Debug" or "All" to obtain more details on what is going on in the server.

It is important to realise that setting the logging level to "Debug" or "All" can result in a large amount of logging, and can also have a significant impact on performance. These levels should only be used to diagnose problems, and the level should then be set back to "Information".

Note that the *Save to local disk* option directly above the panel displaying the log file contents can be used to save a copy of the selected log file to disk.

# How to view the manager log files

The CompleteFTP manager also produces a log file, which may be useful to diagnose any connectivity problems or other issues.

The normal location of the manager's log file is *C:\Users\username\AppData\Local\EnterpriseDT\Complete FTP\Logs* where *username* is the name of the user account that is running the manager. Of course this log file is stored on the machine which is running the manager - which is not necessarily the machine running the CompleteFTP service.

# How to change the log file location

The log file locations can be changed via the following steps. The CompleteFTP manager should still be able to view the log files after changing the location.

1. Open CompleteFTPService.exe.config, which is in the install directory. This may require administrative privileges. This is basically a log4net configuration file, and you can change the location of log files there.
2. Make a copy before modifying it!
3. Modify

   %folder{CommonApplicationData}\\Enterprise Distributed Technologies\\Complete FTP\\logs\\Diagnostics.log

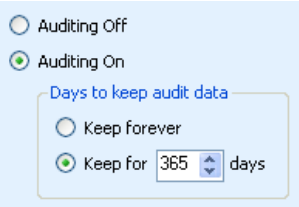
   and similar paths to set your desired location, e.g.

   D:\\CompleteFTP\\logs\\Diagnostics.log
4. Restart the CompleteFTP service to reread the configuration file.

CompleteFTP should now be logging to your new log files.

# How to enable auditing

CompleteFTP supports the use of an audit log. The audit log records all important file transfer operations as well as login/logout.




To enable the audit log, go to the *Diagnostics tab* found on the *Monitoring* tab in the manager, and select the "Configure Auditing" link. Select the "Auditing on" radio button.

Once enabled, CompleteFTP will record every operation in an auditing log file. Log files are 'rolled' by day such that each file contains all the operations for a given day. The file 'Audit.log' contains today's operations. Older files are named using the convention 'Audit.log.YYYYMMDD'. The number of days stored may be changed in this panel (or select "Keep forever" to keep all log files).

# How to modify the logging configuration

CompleteFTP's sophisticated logging system is based on log4net. log4net is highly configurable. If modifications are to be made to the log4net configuration then the full log4net configuration should be placed into a file called `LogConfig.xml`, which should be placed in the directory, `C:\ProgramData\Enterprise Distributed Technologies\Complete FTP\Logs`. This will prevent changes from being overwritten when CompleteFTP is upgraded. The default log4net configuration may be found in `C:\Program Files (x86)\Complete FTP\Server\CompleteFTPService.exe.config`. The XML node, >log4net<, and all of its sub-nodes, should be copied into the `LogConfig.xml` file and used as a basis for a new configuration.

### How to log to syslog

It is sometimes a requirement to direct logging output to *syslog*. This is normally a daemon that listens on UDP port 514. This can be done via the following steps.

1. Create the `LogConfig.xml` file as described above
2. Add a new appender section:

```
<appender name="Syslog" type="log4net.Appender.RemoteSyslogAppender">
    <param name="RemoteAddress" value="127.0.0.2" />
    <layout type="log4net.Layout.PatternLayout" value="%-5p %type: %m%n"/>
</appender>
```

   and add an appender reference to the list of existing appenders:

```
<root>
  <level value="ALL" />
  <appender-ref ref="Recent Logging Messages" />
  <appender-ref ref="Recent Error Messages" />
  <appender-ref ref="Syslog" />
  <appender-ref ref="Audit" />
  <appender-ref ref="Config" />
</root>
```

3. Restart the CompleteFTP service to reread the configuration file.

CompleteFTP should now be logging to syslog.

# How to produce clean log files

The best way to troubleshoot problems is to send a 'clean debug log' to support. This is a log file that contains, as far as possible, only information related to the problem at hand. Note that from version 8.0, the server diagnostics file is preferred, as it contains the configuration file as well as logging.

The summary steps for producing a 'clean debug log' are as follows:

1. Set the logging level to Debug (or All, for even more detail)
2. Stop the CompleteFTP service
3. Delete the existing log file
4. Start the CompleteFTP service again
5. Reproduce the error
6. Send support the (zipped) log file

This, of course, assumes that the service can be stopped without impacting users. If the service cannot be stopped, perform steps 1, 5 and 6 only.

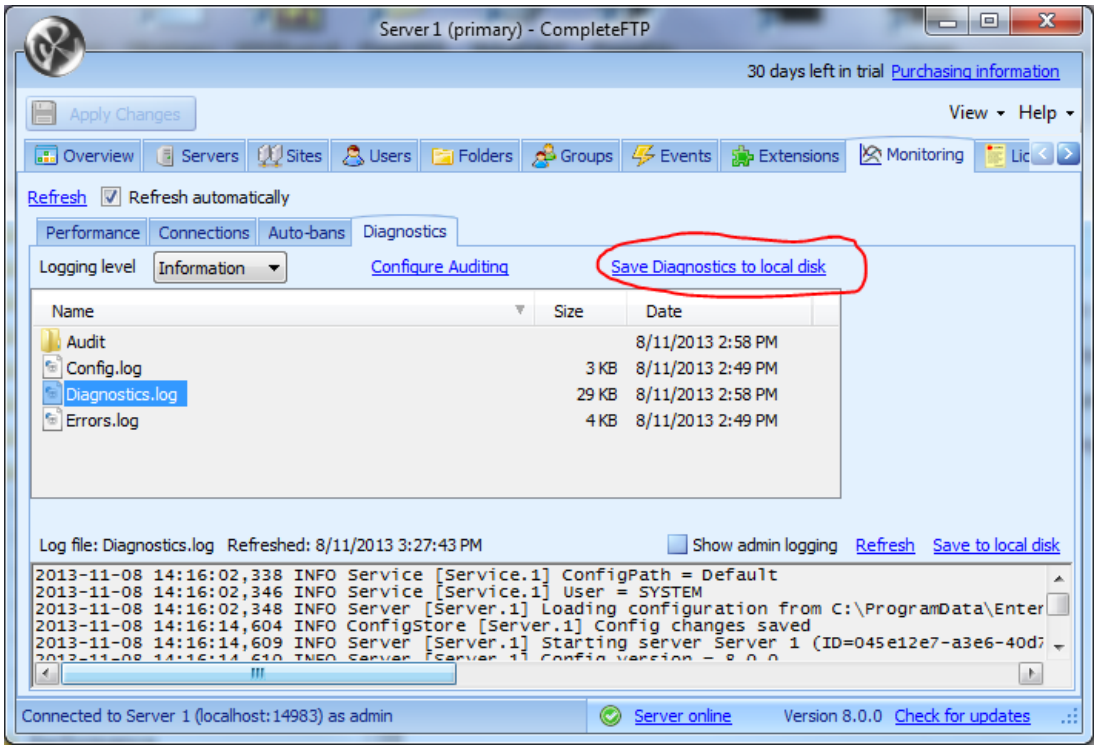
The detailed steps:

1. **Set the logging level to Debug:**
    a. Open CompleteFTP Manager and connect.
    b. Select the Monitoring tab and then select Diagnostics.
    c. Set the Logging level to 'Debug' and click 'Apply Changes'.
    d. Close CompleteFTP Manager.
2. **Stop the CompleteFTP service:**
    a. Open the Services item in the Windows Control Panel.
    b. Select CompleteFTP and hit the stop button.
3. **Delete the existing log file:**
    a. Select 'Log Files' from the CompleteFTP menu group in the Windows start menu.
    b. Find 'Diagnostics.log' and delete (or rename) it.
4. **Start the CompleteFTP service again:**
    a. Open the Services item in the Windows Control Panel.
    b. Select CompleteFTP and hit the start button.
5. **Reproduce the error:**
    a. Do whatever it was that caused the error. E.g. connecting with an SFTP client and getting an error.
6. **Send the (zipped) log file to EDT support:**
    a. Select 'Log Files' from the CompleteFTP start menu (in Windows).
    b. Zip up the Diagnostics.log file and e-mail it to support

# How to get diagnostics for support

From CompleteFTP 8.0, there is a better way to collect diagnostic details to send to support.

1. Open the CompleteFTP manager and go to the Diagnostics tab in the Monitoring panel.
2. Ensure that the logging level is set to Debug (or All).
3. Restart the CompleteFTP service (optional).
4. Attempt to reproduce the problem.
5. Use the "Save Diagnostics to local disk" link to download a zip file containing a sanitised version of your configuration file (without password hashes etc) and your Diagnostics.log file.



## Performance

We are often asked how well CompleteFTP will perform with x users. The short answer is, it depends.

Client applications vary enormously in what they do. Some applications are batch jobs that transfer a single file once per day, while others are transferring every minute. Some applications transfer tiny files, while others transfer large files that can take many minutes, or even hours. The primary constraint on performance will be how many simultaneous transfers are being made at any given time.

CompleteFTP has a low memory footprint so memory should rarely be an issue. For simultaneous transfers, the limiting factors are going to be CPU and bandwidth. CPU is particularly important for encrypted transfers (i.e. SFTP, FTPS and SCP), as encryption is computationally expensive. If CompleteFTP is being run in a VM, we recommend allocating *at least* 2 vCPUs to the VM. If many clients are transferring simultaneously, obviously bandwidth is also going to limit transfer speeds.

If performance is mission critical, it is advisable to simulate the production scenario to ensure that CPU and bandwidth is adequate. It is quite easy to do a basic simulation of many clients - run multiple instances of a test client that continually transfers files back and forth. You may wish to run a few instances on different machines. That should provide a reasonable idea of how well the system is going to perform.

# Administrator Access

CompleteFTP servers can be administered remotely in a variety of ways.

The *CompleteFTP manager* provides the most flexibility, and this guide is primarily concerned with describing its capabilities. The manager connects to the administrator site hosted on the server, and the administrator site settings can be changed as described at this link.

Some administrative tasks can also be performed via the SSH command-line, if the *SSH Terminal* option is enabled for the Admin site. An SSH client is required to log in to port 14983 (the default) with an SSH client. Type 'help' to see what commands are currently available. More details can be found at this link.

Finally, the commands available in SSH can also be scripted, as described at this link.

# How to change the administrator site settings

The *Admin Access* window displays the settings for the Administrator site in the *Standard Edition* and *Professional Edition*. The administrator user name can be set in this window, and the administrator password can also be changed.

More detailed administrator settings can be modified by selecting *Show advanced administrator connection settings*. These settings apply only to the Admin site and are described in the Site/Sites section. For example, the protocol used to connect to the Admin site can be change

In the *Enterprise Edition*, the Admin site is displayed in *Sites* by unchecking the 'Show admin site' checkbox at the bottom left of the *Sites* panel, and so the *Admin Access* panel is not present.

The Admin site is used to administer the server instance. The adminstrator connects to the Admin site via the Complete FTP Manager. By default the Admin site listens on port 14982 for FTP/FTPS, and port 14983 for SSH/SFTP. From version 6.1, the default protocol is set to SFTP, as it is a more secure protocol than FTP.

Note that the SSH Terminal option must be enabled for the Admin site if SSH access is required for command-line administration as described at this link.

# How to add extra administrator users
**(requires Enterprise Edition)**

Often, users want to add extra CompleteFTP administrators, so that they have several accounts with administrator access. This is only possible in the Enterprise Edition.

It is very straightforward to add local admin users, i.e. CompleteFTP users that have administrator privileges. Use the *Add user* link in the *Users* tab of the CompleteFTP manager, and select the *Admin User* menu item.

If you want an existing Windows user to be set as an administrator, it is more complicated. The steps are shown below:

1. Add the Windows user to CompleteFTP
2. Set the home directory of the user on the Admin site to /Admin (see note below)
3. Add the user to the *admins* group

In order to do step 2 you must add an additional site first. This is necessary because the dialog that allows you to set the home directory of a user will only show multiple sites if there is more than one non-admin site. This is by design, for reasons of simplification for the 99% use-case, though in this case it's obviously undesirable.

## How to do command-line administration
**(requires Enterprise Edition)**

In the Enterprise Edition, some administration tasks can be performed using the SSH command-line. The Admin site by default listens on port 14983 for SSH/SFTP. Note that the *SSH Terminal* option for the Admin site must be enabled for SSH clients to connect.

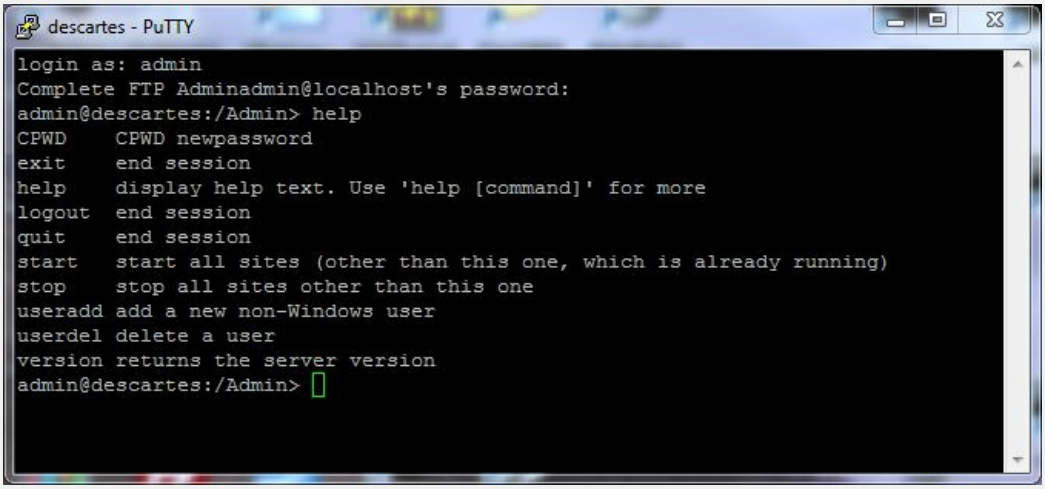
Any command-line SSH client can connect to this port (14983) using the admin user's credentials and use the available commands. On Windows, PuTTY is a good free option for an SSH client. On Unix systems, command-line SSH clients are usually freely available and installed by default.

Once the terminal window is shown after connecting, type 'help' to see what commands are currently available. Type 'help command' to find out more about each command.

There are commands for adding and deleting users, adding and removing folders, and for controlling permissions, all detailed here.

Multiple commands can also be placed into a script and run on the server as described here.

An example of using PuTTY to access the Admin site is shown below:


descartes - PuTTY
login as: admin
Complete FTP Adminadmin@localhost's password:
admin@descartes:/Admin> help
CPWD CPWD newpassword
exit end session
help display help text. Use 'help [command]' for more
logout end session
quit end session
start start all sites (other than this one, which is already running)
stop stop all sites other than this one
useradd add a new non-Windows user
userdel delete a user
version returns the server version
admin@descartes:/Admin>

## How to do script-based administration
**(requires Enterprise Edition)**

In the Enterprise Edition, administrative commands can be scripted, i.e. they can be saved to a file and the contents of the file can be executed as a script on the administrative site. For example, this could be used to add many users at once. This would be difficult to do using the CompleteFTP manager, and still tedious via the command-line. Instead, the script could be generated and then executed in one command.

Like command-line administration, scripts must be run with an SSH client that can run scripts remotely. On Windows, a popular free tool is plink, part of the PuTTY suite of free SSH tools.

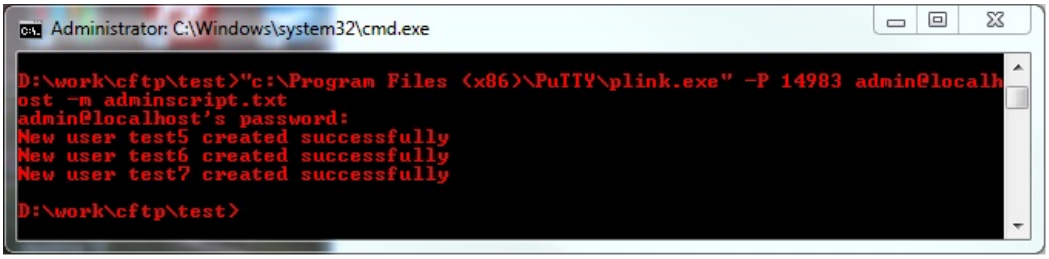
The example below shows how to run a script on the Admin site using *plink*. Note that the port number of the Admin site (default 14983) must be specified via the -P option:

```
plink.exe -P 14983 admin@localhost -m adminscript.txt
```

For example, the contents of a script that adds three users, and then adds a Windows folder mounted as "/downloads" to the virtual file system that belongs to user5 is shown below:

```
useradd -c "user 1" -n user1 -r password test5
useradd -c "user 2" -n user2 -r password test6
useradd -c "user 3" -n user3 -r password test7
folderadd -t windows -v /downloads -p d:\\downloads -u user1 -g users
```

When this script is run using *plink*, the password is prompted for, and the script is run on the Admin site of the server, as shown below. The scriptfile in this example is *adminscript.txt*:



# Administrator command-line

**Folder commands**

| Command | Description | Usage |
|---|---|---|
| chgrp | Change the group of a folder in the virtual filesystem | chgrp group virtualpath (e.g. chgrp mygroup /test) |
| chmod | Change the permissions of a folder in the virtual filesystem. Uses Unix-style permissions. | chmod permissions virtualpath (permissions in Unix format, either octal or rwx, e.g. chmod u=rwx,g=rx,o=r /test) |
| chown | Change the owner of a folder in the virtual filesystem | chown username virtualpath (e.g. chown myuser /test) |
| folderadd | Create a new folder in the virtual filesystem | folderadd -t virtual\|windows\|networkmacro -v virtualpath [-p path] [-u username] [-g groupname] |
| folderdel | Delete an existing folder in the virtual filesystem | folderdel -v virtualpath |
| foldermod | Modify an existing folder in the virtual filesystem | foldermod -v virtualpath -p pathtochange |

**User commands**

| Command | Description | Usage |
|---|---|---|
| useradd | Add a new non-Windows user. Optionally, the MD5 hash of the password can be set rather than using the raw password (which will be converted to a hash). | useradd [-h] [-c comment] [-n fullname] [-d home_dir] [-p hashedpassword\|-r rawpassword] username |
| userdel | Delete a user. This cannot be undone! | userdel username |

**Site commands**

| Command | Description | Usage |
|---|---|---|
| start | Start all sites (other than the admin site, which is required to be running to access this command) | start |
| stop | Stop all sites (other than the admin site, which is required to be running to access this command). Use with caution, as calling it will stop all public sites. | stop |

**General commands**

| Command | Description | Usage |
|---|---|---|
| help | Display help text for commands | help commmand (e.g. help useradd) |
| logout | End current administrator session | logout |
| quit | End current administrator session | quit |
| version | Returns the current server version | version |

## Security overview

Security is a never-ending battle when it comes to servers - there is always the threat of a malicious attacker gaining access to your systems and causing damage.

Fortunately, there are a number of steps that can be taken as detailed in how to secure your server that will minimize the possibility of your CompleteFTP server being compromised.

The most important of these is to keep your software up to date with the latest patches (both CompleteFTP and Windows), and to actively monitor your system for signs of intruders.

# How to disable SSL 3.0

The world-wide standard for securing TCP connections began as the Secure Sockets Layer, or SSL. SSL 3.0 was succeeded by Transport Layer Security (TLS) in 1999, and together they are often known as SSL/TLS. Servers have generally kept SSL 3.0 available as an option so that older secure clients can connect to them without supporting TLS, but vulnerabilities such as POODLE mean that SSL 3.0 support is a security risk that should be avoided.

SSL 3.0 support is disabled by default from version 8.1.3 of CompleteFTP, so upgrading to 8.1.3 is highly recommended. SSL 3.0 can still be enabled if required, but it is not recommended unless HTTPS is not being used (i.e. HTTPS is disabled and only FTPS is being used). POODLE is not easy to exploit via FTPS, and so the risk is not high if HTTPS is not enabled.

# Licensing overview

The *Licensing* window displays licensing details for the CompleteFTP server installation that the manager is currently connected to.

The *Edition* displays the product edition that has been purchased, either the *Standard*, *Professional* or *Enterprise* Edition. Some features are only available in the Professional or Enterprise Editions. The differences between the editions are shown here.

The expiry date displays the date that the server license expires. Trial installations are given 30 days while production licenses normally display *Never*.

# Migrating to another machine

At times it may be necessary to migrate a CompleteFTP installation to another machine. To do so while preserving settings such as user logins, passwords and home folders, follow the steps below:

1. Copy the files and subdirectories of the data-directory (see below) from the old machine to the same location on the new machine.
2. Install CompleteFTP on the new machine. You **must** choose to preserve the existing configuration when given this option. A later version may be installed on the new machine as the installer will upgrade the configuration.
3. Contact support to request an additional activation key (unnecessary for corporate licenses with unlimited keys).
4. Reactivate the new installation (the activation key is tied to the Windows installation, so reactivation is required).

**Locating the data directory**

On Windows 7, Vista and Server 2008 the data-directory is:

*C:\ProgramData\Enterprise Distributed Technologies\Complete FTP*

On Windows XP and Server 2003 the data directory is:

*C:\Documents and Settings\All Users\Application Data\Enterprise Distributed Technologies\Complete FTP*

# How to resolve restricted mode

If CompleteFTP detects an invalid or missing license, it goes into *restricted mode*. In this mode it only permits local connections (i.e. connections originating from the same machine it is installed on).

This problem is generally because of a migration to a new machine. It is also possible if CompleteFTP is running in a virtual machine that the machine's ID has been changed. In both these scenarios the machine ID encoded in the license does not match that of the new machine.

The resolution is to activate the installation. This necessarily uses up one of your purchased licenses. If you are moving your CompleteFTP installation, you can contact support to request an additional license be allocated.

# How to automatically import a license

Sometimes it can be extremely useful to be able to automatically import an activation key into a CompleteFTP configuration file (config.sdf).

For example, this may be required in a failover scenario. If CompleteFTP is set up on a primary machine, and a secondary machine maintains a backup, administrators may choose to regularly copy config.sdf across to the failover machine.

However the primary machine's config.sdf file will contain the activation key for the primary machine. The activation key for the failover machine must be inserted in config.sdf for CompleteFTP to operate with this configuration file on the failover machine.

This is easily accomplised by entering the activation key via *Enter Activation Key* in the Licensing panel. However this is a manual process, and it may be desirable for the process to be automated.

Updating the activation key automatically can be accomplised with the *license* utility that can be found in the Server directory of the installation (typically C:\Program Files\Complete FTP\Server). This utility permits importing or exporting of the activation key to or from config.sdf:

*license -c config.sdf [-i infile|-e outfile|-d] [-h]*

| | |
|---|---|
| -c config.sdf | : CompleteFTP configuration file |
| -i infile | : file to import activation key from |
| -e outfile | : file to dump activation key to |
| -d | : delete the activation key from the database |
| -h | : show this help |

For example, to import an activation key saved in a text file:
*license -c C:\ProgramData\Enterprise Distributed Technologies\Complete FTP\config.sdf -i license.txt*

Note that this tool will not work with CompleteFTP clustered servers (as replicating the configuration is the task of the cluster). The Enterprise Edition is required for cluster support. You can contact us to upgrade to the Enterprise Edition from the Standard or Professional Editions if you wish to access this functionality.

## Support

The Enterprise Distributed Technologies website offers a rich question & answer board for discussion of the CompleteFTP server. It is fully searchable and contains advice and answers to many issues that administrators have faced when using the server.

The Frequently Asked Questions should also be consulted before contacting support.

# How to contact support

Customers and those trialing a product may contact support. Before doing so, please check the site resources, as your question may be answered there.

When contacting support with a problem, always include the following:

- the section of the server log file that illustrates the problem (please zip the log file). Please ensure that the log level is set to "All" - see logs. Ideally, you should get a server diagnostics file if possible.
- any relevant information or logging from a client application if it is having problems connecting. Ensure the maximum log level is being used.
- the version of CompleteFTP you are using, if that is not shown in the log file

Please include as much information as possible; it will save you the time that it takes us to ask for it.

Please check your spam folders for our replies if you haven't seen a reply from us within 24 hours.

## Frequently Asked Questions

Frequently Asked Questions are an excellent source that should always be consulted prior to contacting support

## CompleteBox revision history

| Version | Changes |
| --- | --- |
| Version 1.0.0.3<br>(27 December 2013) | • Fixed Reshare bug. If resharing after an idle period, the interface would be locked and displaying a message about retrying to connect to server. |
| Version 1.0.0.2<br>(12 December 2013) | • Fix CompleteBox right-button click from Windows Explorer stopped working bug. |
| Version 1.0.0.1<br>(2 December 2013) | • First release of file-sharing client CompleteBox, designed to work with CompleteFTP server. |

# Protocol Overview

FTP (File Transfer Protocol) is a long established Internet standard designed to transfer files (and information about files) across computer networks.

FTP is defined in the standards document known as RFC959.

FTP requires a client program (an FTP client) and a server program (an FTP server). The client can fetch files and file details from the server, and also upload files to the server. The server is generally password protected.

FTP commands are initiated by the client, which opens a TCP connection called the control connection to the server. This control connection is used for the entire duration of a session between the client and server. A session typically begins when the client logs in, and ends when the quit command is sent to the server. The control connection is used exclusively for sending FTP commands and reading server replies - it is never used to transfer files.

Transient TCP connections called data connections are set up whenever data (normally a file's contents or file listings) is to be transferred. For example, the client issues a command to retrieve a file from the server via the control channel. A data connection is then established, and the file's contents transferred to the client across it. Once the transfer is complete, the data connection is closed. Meanwhile, the control connection is maintained.

See also:

Active and Passive Modes

FTP Commands

Sample Scenarios

Data Types

Session Commands

File Commands

Directory Commands

# Active and Passive Modes

Data connections may be set up in two different ways, active and passive. Note that active and passive refer to the operation of the FTP server, not the client.

**Passive mode**

In passive mode, the client sends a PASV command to the server. This tells the server to listen for a connection attempt from the client, hence the server is passively waiting. The server replies to PASV with the host and port address that the server is listening on. The client deciphers this reply and when a data connection is required, attempts to initiate the connection to the server at this address.

**Active mode**

In active mode, the server actively connects to the client. To set up active mode, the client sends a PORT command to the server, specify the address and port number the client is listening on. When a data connection is required, the server initiates a connection to the client at this address.

Generally the server is responsible for closing data connections.

Next: FTP Commands

# FTP Commands

FTP commands sent across the control connection consist of simple text strings (and follow the Telnet protocol - see RFC 854). For example, to retrieve a file, the client sends "RETR filename" on the control connection to the FTP server. To transfer a file, the client sends "STOR filename".

The FTP server acknowledges each command with an FTP reply, which consists of a three digit number followed by human-readable text. The first digit indicates if the response is good, bad, or incomplete. If an error occurred, the second digit may be used to indicate what type of error occurred. Similarly, the third digit can indicate more details of the error.

The first digit is the most important, and the five possible values are described below:

| Reply | Description |
|---|---|
| 1yz | Positive Preliminary reply. The request action has been initiated, but another reply is to be expected before the client issues another command. |
| 2yz | Positive Completion reply. The requested action has successfully completed, and the client may issue another command. |
| 3yz | Positive Intermediate reply. The command has been accepted, but more information is required. The client should send another command in reply. |
| 4yz | Transient Negative reply. The command failed, but it can be retried. |
| 5yz | Permanent Negative Completion reply. The command failed, and should not be repeated. |

Next: Sample Scenarios

## Sample Scenarios

**Example 1**

For example, to change directory the client sends:

```
CWD dirname
```

The server responds with:

```
250 CWD command successful
```

As the reply begins with a '2', we know the command sequence is completed.

However if we attempt to change directory to one that does not exist:

```
CWD nonexistentdir
```

The server responds with:

```
550 nonexistentdir: The system cannot find the file specified.
```

As the reply begins with a '5' we know that the command failed.

**Example 2**

To transfer a text file, we issue a 'RETR' command to the server. However to transfer the file we require a data connection to be set up. This can be done using active or passive mode.

As discussed previously, in active mode, the client sends a 'PORT' command, indicating what address and port number the server should connect to:

```
PORT 192,168,10,1,4,93
```

The first four digits are the IP address, and the last two encode the port number (in two 8-bit fields, the first being the high order bits of the 16-bit port number).

The server responds with:

```
200 PORT command successful.
```

This indicates that the data connection has been established.

Next, the 'RETR' command is issued:

```
RETR abc.txt
```

The server responds with:

```
150 Opening ASCII mode data connection for abc.txt(70776 bytes).
```

The reply begins with a '1', so we know that the command was successful, but the server will be sending another reply – the client cannot issue another command until this is received.

Eventually, the server sends:

```
226 Transfer complete.
```

The command sequence is complete, the file has been transferred, and the client can issue another command.

See RFC959 for details about the second digit, and more extensive examples.

Note that most standard command-line FTP clients support debug mode, which displays the FTP commands that are being sent to the server, and the reply strings that are received back. Typing "debug" at the prompt will usually put the client into debug mode.

Next: Data Types

# Data Types

The two most common data types in usage are ASCII and binary.

ASCII is the default data type, and is intended for the transfer of text files. A line of text is followed by <CRLF>. Note that different operating systems use different end of line terminators.

Binary type (known as IMAGE in FTP) is used to transfer binary files. A byte-by-byte copy is made of the source file. Graphical images, video and executable files are all binary files. If they are transferred as ASCII, they will be corrupted.

Next: Session Commands

## Session commands

To begin an FTP session, the USER command is sent to the server:

```
> USER ftpuser
```

The server responds with:

```
331 Password required for ftpuser.
```

The client must respond with the password:

```
> PASS mypassword
```

The server responds with:

```
230 User ftpuser logged in.
```

The session is now established, and the user can begin issuing various file and directory- related commands.

To end the session, the client sends:

```
> QUIT
```

The server responds with:

```
221
```

The session is now closed, and any further attempt to send commands to the server will fail.

Next: File Commands

# File Commands

FTP supports numerous file-related commands.

Files can be deleted (DELE) and renamed (RNFR,RNTO) as well as stored (STOR) and retrieved (RETR). When a file is stored, it can be written over or appended to (APPE).

See the Sample scenarios examples for more details.

Next: Directory Commands

## Directory Commands

FTP supports a variety of directory-related commands.

Directories can be created (MKD), removed (RMD), or changed into (CWD, CDUP).

Two types of directory listings can be made with FTP.

The LIST method obtains a list of files (and possibly directories). If a directory is specified, the server returns a list of files in the directory, together with system specific information about the files. The file information sent will vary depending on the server system. The data type should be set to ASCII for this file name list. If no directory is specified, details of the current working directory listing are sent.

The NAME LIST (NLST) method is similar to LIST, but only file names are returned. No other information about the files is sent. Again, the data type should be set to ASCII.

Next: FTPS - Securing FTP with TLS

# FTPS - Securing FTP with TLS

A significant problem with "plain" FTP is that it is not secure - usernames, passwords and data are sent across the network in the clear. This means eavesdroppers sniffing the network will have little trouble obtaining credentials and copies of the transferred files.

FTPS was designed to fix this by encrypting communications using the SSL/TLS protocol, which is specifically designed for securing network connections.

FTPS is an IETF standard as described in RFC 4217.

See also:

- Implicit FTPS and Explicit FTPS
- Securing Control and Data Channels
- FTPS Commands
- FTPS Usage

## Implicit FTPS and Explicit FTPS

An early version of FTPS is now known as *implicit FTPS*. Upon connecting, implicit FTPS clients automatically start securing the connection using SSL/TLS.

This is a problem for unencrypted FTP clients - they will no longer be able to connect on a server port that requires securing the connection immediately. If FTP and FTPS (implicit) are to be supported on the same server, they require different port numbers. Typically, implicit FTPS uses port 990 rather than the standard FTP port 21.

Explicit FTPS fixed this by requiring the AUTH command to be sent by the client prior to securing the connection. This meant that unencrypted FTP clients can connect on the same port as FTPS clients - the unencrypted clients simply never send an AUTH command and the session remains unencrypted. FTPS clients send the AUTH command prior to logging in so that credentials are secured.

Implicit FTPS clients can still be found, but explicit mode is always preferred. Unless implicit mode is required it is best to have implicit mode FTPS disabled in CompleteFTP.

Next: Securing Control and Data Channels

## Securing Control and Data Channels

FTP sessions use two channels: *control* and *data*. Only one control-channel is used in each session, but several data-channels may be used – one for each data transfer. The AUTH command only secures the control-channel. Data-channels are not secured until PBSZ and PROT commands are issued. These commands tell the server whether or not subsequent data-channels should be secure.

Clients can connect to FTPS servers in unencrypted mode, and then switch to secure mode upon request. To do so, the client issues an AUTH command, upon which the client and the server negotiate a secure connection. After the switch, all FTP commands are encrypted, but importantly, data is not encrypted unless further commands are provided.

Next: FTPS Commands

## FTPS Commands

Three commands are used, AUTH, PBSZ, and PROT. One of these, PBSZ, appears to be redundant and is probably included only to satisfying the RFC specification.

**AUTH (AUTHentication)**

The AUTH command takes a single parameter which defines the security mechanism to be used, usually 'SSL' or 'TLS'.

```
AUTH TLS
```

WIth this command, an attempt is made to negotiate a TLS connection on the control channel.The server tries to validate itself to the client by sending its certificate (Server Validation). It may also involve the client sending its certificate to the server (Client Validation).

**PBSZ Protection Buffer SiZe)**

The PBSZ command is intended to define the buffer-size to be used by the security mechanism when it is encrypting data on the data-channel. However for TLS this setting is redundant and a value of '0' is always passed as a parameter.

```
PBSZ 0
```

While this call is redundant, it is required and must precede the PROT command.

**PROT (data channel PROTection level)**

PROT defines whether or not the data channel is to be protected or not. Either the data channel is *Clear* (the default), or *Private*. Clear means that no security is used on the data-channel (meaning files are transmitted without encryption), and Private means that the data-channel should be encrypted. So there are two possible PROT commands:

```
PROT C
```

for an unsecured data-channel, and

```
PROT P
```

for an encrypted data-channel.

Next: FTPS Usage

## FTPS Usage

A typical explicit FTPS session might consist of the following sequence of commands:

| | |
|---|---|
| `> USER (user-name)` | *Log in* |
| `> PASS (password)` | |
| `> LIST` | *Get a directory listing* |
| `> AUTH TLS` | *Switch to TLS on control-channel* |
| `> RETR (file-name)` | *Download a file (without security)* |
| `> PBSZ 0` | *Switch to TLS on the data-channel* |
| `> PROT P` | |
| `> STOR (file-name)` | *Upload a file (with security)* |
| `> QUIT` | *End session* |

In this example, the first three commands (USER, PASS, and LIST) are standard FTP and therefore insecure. The AUTH command causes the rest of the commands to be sent to the server securely, in other words, an attacker cannot see which commands are issued. The RETR command (to fetch a file from the server), being after AUTH, is protected, but the actual file that is transferred is not protected since it precedes the PBSZ and PROT commands. PBSZ and PROT tell the server to use TLS on all future data-channels, thus the file transferred in the STOR command (which stores a file on the server) is secure.

**Rules**

There are two rules regarding the issuing of explicit FTPS commands that must be followed:

1. AUTH must precede PBSZ
2. PBSZ must precede PROT

Apart from these, a FTPS server has policies regarding access permissions to its resources. These policies will also determine the order in which commands must be issued. There are too many possible policies to list here, but a few examples of such policies are given below along with their consequences in terms of the issuing of commands.

| ***Policy*** | ***Consequences*** |
|---|---|
| • No unprotected commands | AUTH must be issued before any other commands. |
| • Certain users are not permitted to log in without security. | The USER command is rejected for particular users unless preceded by a successful AUTH command. |
| • No unprotected data may be transferred | A 'PROT P' command (preceded by a PBSZ command) must be issued before any files are transferred. |
| • Allow TLS authentication instead of USER/PASS authentication | A client certificate must be supplied and USER/PASS commands are not required. |

Next: The Essentials of FTP Security

# The Essentials of FTP Security

While users of FTPS may safely remain ignorant of most of the underlying security mechanisms, it is necessary that they understand the basics of public keys and certificates. In some cases, it is also necessary to understand which encryption/decryption algorithms (or *ciphers*) are available and what level of protection they offer.

See also:

- Public Key Cryptography
- Certificates and Certificate Authorities (CAs)
- Obtaining Keys and Certificates
- Server and Client Validation
- Hostname Checking
- Selecting Ciphers

# Public Key Cryptography

Public Key Cryptography (PKC) is a paradigm which uses a pair of keys in a given communication; one key is used for encrypting the message and the other key is used for decrypting it. Each key may serve either purpose, but a message encrypted using one key may ONLY be decrypted by the other key.

The following illustration demonstrates how such a pair of keys may be employed for secure communication:

> *Party A has generated a key-pair. They retain one key, the **private key**, and distribute they other key, the **public key**, to Party B in a trustworthy way (see Section 4.1.2).*
>
> 1. *A encrypts a message using the private key and sends it to B. If B is able to decrypt the message using the public key, then B may be confident that the message did in fact originate from A, since only A has the private key.*
> 2. *B encrypts a message using the public key and sends it to A. Since the message may only be decrypted using the private key, and only A has this key, B may be confident that only A will be able to read the message.*

Thus, using *A*'s private/public pair of keys, *B* can ensure that (1) *A* is who they purport to be, and (2) Any messages that are sent to *A* can be read by that party only. However, the following weakness remain: (1) *A* cannot be confident that *B* is who *B* purports to be, and (2) Communications from *A* to *B* may be read by anyone who has the public key.

While both of these weaknesses may easily be overcome if *B* has their own key-pair and has provided the public key to *A*, this is often not practical due to the amount of effort involved. However, the latter shortcoming may easily be overcome in the following way:

> 3. *B automatically generates a temporary key-pair. Since B may be confident that its messages to A are only read by A, B can securely provide one of the keys to A. Once A has received this key, they may use it to encrypt any messages sent to B. They may therefore be confident that only B can read any subsequent messages.*

Thus, a single private-public key-pair has the potential to offer the following security:

I. Parties receiving messages from the owner of the key-pair can verify that encoded messages originated from the owner.

II. Secure messages may be sent between the owner and other parties.

As mentioned, this presumes that the owner of the key-pair is able to distribute their public keys in a trustworthy manner. In practice, this is achieved by means of public key certificates and Certificate Authorities.

Next: Certificates and Certificate Authorities (CAs)

## Certificates and Certificate Authorities (CAs)

The previous section emphasized the need for public keys to be distributed in a trustworthy manner. Simply sending the key electronically should not be considered safe as the message might be tampered with on the way, though this is done in many cases. Having one person physically hand another a memory-stick containing the key would be probably acceptable, but is usually impractical. The solution used on the Internet is to use a trusted third party called a Certificate Authority (CA).

A CA is an organization which specializes in issuing public key certificates. They only issue certificates to parties (or *subjects*) after they have provided sufficient documentary evidence of their identity. There are only a few CAs in the world and, since their viability relies on their trustworthiness, they can usually be relied on to do a good job of validating their subjects.

Each CA has its own private-public key-pair and its own certificate (called a *root certificate*). Since there are so few CAs and since they rarely change their keys, it is feasible for software to be distributed with a list of the certificates of all existing CAs. For example, Microsoft's and Netscape's browsers are both distributed with files containing lists of root certificates.

A certificate issued to a subject by a CA contains:

- the identification of the subject,
- the public key of the subject,
- the identification of the CA which issued the certificate, and
- the digital signature, provided by the CA, which verifies the contents of the message.

A subject's certificate may be validated in the following way:

1. *Use the identification of the CA to find the appropriate root certificate.*
2. *Use that root certificate to check the digital signature and thereby prove that the CA issued the certificate and that the information in the certificate has not been tampered with.*
3. *Verify the identification of the subject.*

Once this has been done, the party can trust that the public key in the certificate is indeed the public key that they expected. This public key may henceforth be used to establish secure communications with the subject in the manner described in Section 12.

Note that although the CA has signed a certificate containing the public key of the subject, the CA does not have the subject's private key. To issue a certificate, the CA only requires access to the public key. The CA obtains this public key from the subject applying for a certificate. It is therefore up to the subject to generate the private-public key-pair before applying to a CA for a certificate.

Next: Obtaining Keys and Certificates

## Obtaining Keys and Certificates

A private-public key-pair is easy to generate. Likewise, an “uncertified” certificate may be generated with little effort, but obtaining a trustworthy certificate from a CA necessarily requires some work, time, and (usually) cost. It involves interacting with the CA to prove identity, and waiting for the CA to digitally sign the certificate.

Alternatively OpenSSL may be used. The reader is referred to the OpenSSL Key HOWTO for instructions on generating key-pairs and to the OpenSSL Certificate HOWTO for generating certificates. It is recommended that keys with a length of at least 768 bits are used. OpenSSL may be obtained from http://www.openssl.org/source/ or http://www.openssl.org/related/binaries.html.

Next: Server and Client Validation

## Server and Client Validation

The previous sections described how public key certificates may be used to validate parties involved in secure communications. They also explained why some work, time, and cost is involved in obtaining certificates.

The nature of Internet usage is such that it is important to distinguish between server validation and client validation. Server validation allows a client to know that it is talking to the intended server. Conversely, client validation allows a server to know that it is talking to the intended client.

It is often more important for a client to know that it is talking to the intended server than the converse. The reason for this is often financial. For example, if a client is purchasing an item from an online retailer, the client needs to be certain that their credit card details are going to the intended destination. While it might be nice for the retailer to be certain where the money if coming from, it is not usually essential. Therefore, server validation is nearly always used in such transactions, but client validation is less often used. Other applications, such as Internet banking often use both client and server validation.

In FTPS, both server and client validation by certificate are optional. Though the server's certificate is always sent, it is up to the client whether or not it validates the certificate. It is up to the client whether or not it will try to validate itself to the server, but some servers have a policy of not allowing unvalidated clients to access some or all its resources.

It is important to note that, although many FTPS servers don't request client certificates, most require a user-name and password to be sent. If these are sent over a secure control channel then a reasonable level of client validation is inherent.

Next: Hostname Checking

# Hostname Checking

One of the steps in server validation is host-name checking. Host-name checking is a simple check that is performed when a secure connection is being established. It involves comparing the following two items:

- the host-name that is (usually) contained in the certificate which the FTPS server is presenting
- the host-name of the machine on which the FTPS server is running

If they match then one can be confident that the server to which the client is connected is in fact the server to which the certificate was issued. If they do not match, then there's a possibility that the certificate has been stolen and that the server, to which the client is connected, is attempting to "impersonate" the actual server to which the client is actually connected. This is a form of "man-in-the-middle" attack, which gives the attacker complete control over the data being sent and received.

Unfortunately, the most widely compatible version of the X.509 certificate standard does not specify exactly how a host-name should be defined within a server certificate. The convention is that the Common Name (CN) field of the certificate should be used, and, while this is followed by the majority of Certificate Authorities (CAs), it is not universal.

If it is possible to configure the FTPS server's certificate then the Common Name (CN) field of the certificate must be the same as the host-name of the machine on which the FTPS server is running.

Disabling host-name checking is strongly discouraged and should only be done as a last resort if the FTPS server's certificate cannot be configured so that its CN parameter contains its host-name.

Next: Selecting Ciphers

## Selecting Ciphers

SSL and TLS can use a variety of encryption/decryption algorithms (called *ciphers*). In a single secure connection, as many as four different ciphers may be used for various purposes; this set of ciphers is called a *cipher suite*. Each party in a secure connection must designate which cipher suites it is going to support. When a new secure connection is made, the parties involved try to agree on which cipher suite to use. There must be at least one cipher suite that is available on both sides of the connection for this to be possible.

Different cipher suites have different levels of security and performance. The lower the level of security, the easier the cipher is to break. Unfortunately, stronger ciphers usually offer slower performance. Hence, there is a certain level of trade-off between the two. For this reason, the decision on which cipher suites to support is left to the developers and/or users of SSL and TLS applications.

Every cipher suite has a standard name (e.g. TLS_RSA_WITH_RC4_128_SHA). This name reveals which ciphers are used in the suite.

Some guidelines which may be useful when selecting cipher suites are:

- Triple DES offers high security but relatively poor performance (look for names with the characters, 3DES).
- DES (not 3DES) can be cracked relatively easily
- 128-bit RC4 offers high performance and reasonable security (look for names with the characters RC4_128).
- SHA is preferable to MD5 (choose algorithms whose names ending with SHA ahead of those ending with MD5).
- Export version of algorithms are deliberately weakened; avoid them if possible (i.e. avoid 40-bit suites).

# SFTP and SCP

SFTP is an abbreviation for SSH File Transfer Protocol, and is exactly that - a protocol for transfering files over an SSH connection.

SFTP is **not** the standard FTP protocol running over SSH. Although SFTP has similar capabilities and even similar commands to standard FTP, these similarities are purely superficial. The protocol is completely different and incompatible with FTP and its secure extension, FTPS.

SCP is also a file transfer protocol that runs over SSH connections. It is a precursor to SFTP, and allows the copying of files and directories over SSH.

In order to understand SFTP and SCP, it is helpful to have a basic understanding of SSH.

See also:

- SSH - Secure Shell
- SCP - Secure Copy
- SFTP - SSH File Transfer Protocol
- Comparison of FTPS and SFTP

# SSH - Secure Shell

SSH is a standard designed to allow logging in and execution of commands on a remote computer in a manner similar to telnet, rlogin, and rsh. Unlike these protocols, it does this through an encrypted network connection thus offering a much higher level of security.

The first version of the standard, SSH-1, was designed in 1995 by Tatu Ylönen. The second version, SSH-2, is being standardized by the IETF SECSH working group. It offers a higher level of security than its predecessor.

In order for a computer to be able to accept SSH connections, it must be running an SSH server, such as *sshd*, on a publicly accessible port (usually port 22). The client computer must have an SSH client, such as *CompleteFTP*, and be known to the server.

Private/public key-pairs in SSH typically use either the DSA or RSA asymmetric key algorithms. Most SSH servers support both.

Clients perform server validation in SSH via a known hosts file. The client maintains a file containing the hostname (or IP address) of the SSH server, together with the server's public key. When clients connect to the server, they are sent a copy of the server's public key which they can compare with their own record of the server's public key.

The server authenticates clients who connect to it. The client must be previously set up as an SSH user (or in some cases as a user on the server machine). In password authentication, the client supplies their password which the server authenticates as belonging to that user. In public key authentication, the client uses its private key to sign some data, and sends the signature to the server. The server uses the client's public key to verify the signature. In this case the client's public key must be available on the server.

# SCP - Secure Copy

Since the early days of SSH, file transfers have been supported through a command called SCP. This command simply securely copies files or directories between remote computers. It provides no other file operations such as listing, deleting, renaming, and directory navigation.

SCP itself does not provide authentication and security - it relies on the underlying protocol, usually SSH.

CompleteFTP offers full support for SCP.

# SFTP – SSH File Transfer Protocol

The SFTP protocol was developed as a solution to the limitations of SCP. It provides all the operations that have come to be expected with FTP, and is generally preferable.

SFTP does not provide authentication and security itself - it relies on the underlying protocol, usually SSH. The two protocols are commonly used together - so much so that SFTP generally means that it is running over SSH.

CompleteFTP offers full support for SFTP.

# Comparison of FTPS and SFTP

While FTPS and SFTP are completely different protocols, they offer the same basic feature – secure file transfers. It is therefore common to be faced with the choice of one or the other. This section provides some pros and cons of these two protocols.

**Security**

Under ideal conditions SFTP and FTPS are able to offer comparable levels of security, but many SFTP deployments suffer from a vulnerability that is an artifact of SFTP's close relationship with SSH. The problem arises when you want to allow client SFTP access on a server but not SSH access. This is generally not a problem for pure SFTP servers (such as CompleteFTP), but for SSH/SFTP servers such as OpenSSH it can be quite complex and error-prone. So if you are not very careful when you set up your servers, users on machines with the SFTP client installed will be able to use an SSH client to log into the server and execute commands. This is not a problem with FTPS since this is purely a file transfer protocol and not a remote console protocol.

**Upgrading**

FTPS is a straight-forward extension to an existing FTP infrastructure. It is supported by most commercial servers and many open source servers (e.g. wu-ftpd and proftpd), so enabling FTPS on a server is usually just a matter of adding a few configuration options. There is no need to run additional servers since FTPS servers invariably also support FTP. There is also no need to open additional ports in firewalls since FTPS uses the same ports as FTP. It is important to note that data-transfer problems can sometimes arise when changing from FTP to FTPS - see "Firewalls" section below.

**Certificates**

SFTP uses keys rather than certificates. This means that it can't take advantage of the "chains of trust" paradigm facilitated through Certificate Authorities. This paradigm makes it possible for two entities to establish a trust relationship without directly exchanging security information, which is important for some applications. FTPS uses certificates and therefore can take advantage of this paradigm. SFTP clients must install keys on the server.

**Firewalls**

SFTP often works better through some firewalls since it does not rely on multiple connections like FTP does. As explained in an earlier chapter, FTP and FTPS both use a control channel to send commands, and a new data connection for each file transfer. While the control channel is usually easily connected, it is common to experience firewall-related problems when connecting data-channels. This is particularly so in FTPS where the FTP-specific features of most firewalls are ineffective due to encryption. Since SFTP relies on a single network connection, it does not suffer from these problems.

# HTTP Protocol Overview

HTTP (HyperText Transfer Protocol) is a well established Internet protocol designed to transfer content (e.g. HTML files and images) across networks. HTTP is defined in numerous RFCs (e.g. RFC 2616), which can be obtained from the Internet Engineering Task Force.

HTTP is a request-response protocol. A client such as a Web browser sends an HTTP request to an HTTP server, which is hosting content such as HTML files and images. The server sends a response message to the client, which will generally include the content the client requested if it is available. The content (or resources) hosted by the HTTP server is identified by Uniform Resource Locators, or URLS, using the http or https notation. .

The original version of HTTP (HTTP/1.0) used a separate TCP connection to the server for every request-response. This meant that HTML pages with images required a connection for the page as well as for each image. This was not easily scalable, and HTTP/1.1 can reuse connections multiple times.

See also:

- HTTP methods
- HTTP sessions
- HTTPS

## HTTP methods

HTTP defines various request methods that clients can send to HTTP servers. Generally they involve performing an action on the specified resource. The most commonly used request methods are listed below:

| Method | Description |
|---|---|
| GET | Retrieve the specified resource (usually an HTML page or file). |
| HEAD | Same as GET, except only retrieve the response header, not the body (i.e. the contents of the requested resource). |
| POST | Submit data to be processed. The data is in the body of the request, and usually comes from an HTML form. |

Next: HTTP sessions

## HTTP sessions

HTTP is a stateless protocol. This means the server is not required to maintain state for a user in between a user's HTTP requests. One common solution is to use HTTP cookies, which are text strings stored by a user's web browser. Typically, a session identifier provided by an HTTP server is stored in a cookie. This cookie is then sent with every request to the server, and the server is able to match the session id with stored state for the user. Session identifiers normally expire within a few minutes, hours or days, depending on the server settings.

Next: HTTPS

## HTTPS

HTTP is not a secure protocol, in the same way that FTP is not a secure protocol - all data is transmitted unencrypted, including usernames and passwords. The HTTPS (HyperText Transfer Protocol Secure) protocol was introduced to provide a way to securely send data via HTTP. It is identical in syntax, but uses SSL/TLS to encrypt the data being sent to and from the server.

HTTPS is especially useful for encrypting usernames and passwords that are being sent to an HTTP server, as well as sensitive individual and corporate data such as credit cards or other payment transactions.

## About this manual

Enterprise Distributed Technologies would like to thank Mark James for making his Silk Icons available with a Creative Commons Attribution 2.5 License. We have used these icons in this manual and highly recommend them.

# Authenticator Class Reference

**Class: Authenticator**

Authentication extensions must extend this class. Only `CheckUserName` and `Authenticate` must be overridden. The only job of `CheckUserName` is to set `userInfo.IsValidUserName` to `true` if the user-name (`userInfo.UserName`) is valid. `Authenticate` must set `userInfo.IsCorrectPassword` to `true` if the password (`authInfo.Password`) is correct for the given user-name (`userInfo.UserName`) . Alternatively, `userInfo.IsValidKey` must be `true` if public key authentication is being attempted. If the home folder of the template user (usually `defaultExtension` - see below) is set to `%ExternalHomeFolder%` the value of `HomeDirectory` should be set to the path of the Windows directory that you wish to be the home of the user.

By default, users that are authenticated by authenticator extensions use the `defaultExtension` as a 'template user'. This allows the administrator to set the properties (e.g. permissions) of these users using the CompleteFTP Manager. The user that is used as a template can be changed by overriding the `TemplateUserName`. This property should return the name of the user that is to be used as a template.

The value of the Configuration field entered in the Manager is accessible via the `ExtensionConfiguration` property.

`LoadedUserInfo LoadUserInfo(IUserInfo suppliedUserInfo)`
Return an object containing the loaded user details if the user-name (`userInfo.UserName`) is valid. This method is called automatically in the default implementation of `CheckUserName`.

`void CheckUserName(IUserInfo userInfo)`
Set `userInfo.IsValidUserName` to `true` if the user-name (`userInfo.UserName`) is valid.

`void Authenticate(IAuthenticationInfo authInfo)`
Set `userInfo.IsCorrectPassword` to `true` if the password (`authInfo.Password`) is correct for the given user-name (`userInfo.UserName`). Or, if public key authentication is being used, set `userInfo.IsValidKey` to `true` if the key details supplied in the `IAuthenticationInfo` are valid. If the home folder of the template user (usually `default_external` - see above) is set to %ExternalHomeFolder% the value of `HomeDirectory` should be set to the path of the Windows directory that you wish to be the home of the user.

`void Initialize(IPlugInInfo info)`
Perform any required initialization that your extension requires.

`void Dispose()`
Perform any clean-up that your extension requires.

`string TemplateUserName`
Optionally override this property to set a template user other than default_external (see above)

`string ExtensionName`
Name of the extension entered in the CompleteFTP Manager.

`Guid ExtensionID`
Identifier of the extension.

`string ExtensionConfiguration`
Configuration entered for the extension in CompleteFTP Manager.

**Interface: IUserInfo**

`string UserName`
User-name that the client is presenting.

`IPEndPoint RemoteEndPoint`
Holds the client's IP address and port-number.

`bool IsValidUserName`
Set to `true` if the user-name is valid.

`string SiteName`
Name of site that is requesting authentication

`Guid SiteID`
Identifier of site that is requesting authentication. To obtain the ID for a given site you need to look at the log file; the ID is logged each time a site is started.

**Class: LoadedUserInfo**

`string HomeDirectory`
The user's home directory if available.

`List<byte[]> DSAPublicKeys`
The list of DSA public key blobs for this user. Each key is a byte array of the standard OpenSSH or SECSH formats for public keys.

`List<byte[]> RSAPublicKeys`

The list of RSA public key blobs for this user. Each key is a byte array of the standard OpenSSH or SECSH formats for public keys.

`string PasswordHash`
The MD5 hash of the user's password

`string PasswordSalt`
The salt prepended to the user's password prior to the MD5 hash. This can be null if a salt has not been used.

`string Password`
Setter only to set the user's password. This populates the `PasswordHash` field.

`bool MustChangePassword`
Set to true if the user must change their password

`List<string> Groups`
Names of groups of which the user is a member (in addition to those of the log-in-as user).

**Interface: IAuthenticationInfo**

`string UserName`
User-name that the client is presenting.

`IPEndPoint RemoteEndPoint`
Holds the client's IP address and port-number.

`bool IsValidUserName`
Set to `true` if the user-name is valid.

`string Password`
Password that the client is presenting.

`IsCorrectPassword`
Set to `true` if the password is correct for the given user.

`bool MustChangePassword`
Set to true if the user must change their password.

`bool IsValidKey`
Set to `true` if public key authentication is successful for the given user.

`PublicKeyAlgorithm KeyAlgorithm`
Provides the public key algorithm used if this is public key authentication.

`byte[] KeyCheckData`
Provides the key check data if this is public key authentication.

`byte[] SignatureBlob`
Provides the signature blob if this is public key authentication.

**Interface: IPlugIn**

`PlugInName`
Name of plug-in as entered in CompleteFTP Manager.

`PlugInConfiguration`
Configuration entered by administrator.

# CompleteFTP Custom Configurer: cftpconfig

The cftpconfig tool may be used for modifying the CompleteFTP configuration database (`config.sdf`). Whether run with or without options the tool will always update config.sdf (schema and/or data) to the latest version of CompleteFTP (unless it's already up-to-date).

By default `cftpconfig` operates on the configuration file at the default location (i.e. `C:\ProgramData\Enterprise Distributed Technologies\Complete FTP\config.sdf`), but this can be changed using the options.

Please note that as `cftpconfig` operates directly on the configuration file, the CompleteFTP service will need to be restarted to reload the configuration. Until this is done changes will not be able to be seen in the CompleteFTP manager. Normally this tool should only be used when the service is stopped.

**Available options:**

| | |
|---|---|
| /c [file] | Customize installation using SQL commands in this file or read from stdin if path is absent or '-' (ctrl+Z to mark end of input). |
| /f file | Configure the given configuration file instead of the configuration file at the default location |
| /h | Show help |
| /i file | Import the configuration in the given file instead of upgrading the configuration at the default location |
| /o | Create a new configuration file rather than upgrading the existing config.sdf. All existing configuration data will be overwritten. |
| /p password | Set administrator password to this |
| /q query | Execute this SQL query (INSERT, UPDATE or DELETE) on the configuration |
| /r | Generate (and write) random administrator password |
| /v | Verbose - write status messages to console |

The schema of the CompleteFTP configuration database is available upon request. The SQL syntax is that for SQL Server Compact with the addition of the UPDATE-FROM command. The syntax of the UPDATE-FROM command is

```
UPDATE TableA SET FieldA1=AliasB.FieldB1 WHERE FieldA2=value1
FROM TableB AliasB WHERE FieldB2=value2
```

A tool such as CompactView may be useful in viewing your config.sdf file.

# Configuration Database Schema

CompleteFTP stores configuration data in a SQL Server Compact database. Modifying this database directly is not generally advised as it's very easy to render CompleteFTP unusable. It is however possible to do using the cftpconfig tool. EnterpriseDT advises users wishing to perform anything but the simplest property-changes to contact support for advice prior to putting changes into production.

This document describes the schema of the CompleteFTP configuration database. It lists the fields of the tables along with notes on how rows should be added. Note that the constraints on the database do not fully embody the format of the data, so the notes should be read carefully.

**Tables**


- Users
- Node
- SiteUser
- Membership
- Group
- Server
- Site
- ServerSite
- NodeType
- PlugIn
- PlugInType
- Authenticator


## Table: User

***Purpose:*** *Stores all explicitly defined users. This does not include users that are authenticated using authentication extensions, including automatic windows users and external database users.*
***Notes:*** *Create one row for each new user.*

| Field | SQL Type | .NET Type | Required | Default | Comments |
|---|---|---|---|---|---|
| UserID | uniqueidentifier | Guid | yes | | Generate new for each user |
| AuthenticationTypeID | nvarchar(40) | string | yes | | "SERVER" for non-Windows users and "WINDOWS" for Windows users |
| AuthorizationTypeID | nvarchar(40) | string | yes | | "SERVER" for non-Windows users and "WINDOWS" for Windows users |
| Description | nvarchar(100) | string | no | | May be NULL. Only used in event-macro |
| Email | nvarchar(256) | string | no | | May be NULL. Only used in event-macro |
| Enabled | bit | bool | yes | | Usually TRUE |
| ExpiryDate | datetime | DateTime | no | | NULL if no expiry |
| FTPEnabled | bit | bool | no | TRUE | TRUE if FTP is to be allowed; FALSE otherwise. |
| FTPSEnabled | bit | bool | no | TRUE | TRUE if FTPS is to be allowed; FALSE otherwise. |
| FullName | nvarchar(100) | string | no | | May be NULL. Only used in event-macro |
| HTTPEnabled | bit | bool | no | TRUE | TRUE if HTTP is to be allowed; FALSE otherwise. |
| HTTPSEnabled | bit | bool | no | TRUE | TRUE if HTTPS is to be allowed; FALSE otherwise. |
| PasswordEncrypted | image | byte[] | no | | Leave as NULL |
| PasswordHash | nvarchar(100) | string | no | | NULL for Windows users and MD5 (unsalted) for non-Windows users |
| Permissions | nvarchar(3) | string | no | | NULL for full permissions or 555 for read-only |
| QuotaSpeedDownload | bigint | long | no | | NULL or download-speed in bytes/sec |
| QuotaSpeedUpload | bigint | long | no | | NULL or download-speed in bytes/sec |
| QuotaStorage | bigint | long | no | | NULL or storage-quota in bytes |
| SCPEnabled | bit | bool | no | TRUE | TRUE if SCP is to be allowed; FALSE otherwise. |
| SFTPEnabled | bit | bool | no | TRUE | TRUE if SFTP is to be allowed; FALSE otherwise. |
| SSHPublicKeyDSA | image | byte[] | no | | NULL or OpenSSH format DSA public key |
| SSHPublicKeyRSA | image | byte[] | no | | NULL or OpenSSH format RSA public key |
| SSHTerminalEnabled | bit | bool | no | FALSE | TRUE if SSH is to be allowed; FALSE otherwise. |
| SSLCertificate | image | byte[] | no | | NULL or PEM format SSL cert (no private key) |
| System | bit | bool | yes | | Must be FALSE |
| UserName | nvarchar(100) | string | yes | | Must be unique. Must include "MYDOMAIN\" for Windows domain users |
| CreatedTime | datetime | DateTime | yes | | Set only when creating |
| ModifiedTime | datetime | DateTime | yes | | Update every time a change is made |

## Table: Node

***Purpose:*** *Stores folders of the virtual file-system.*

***Notes:*** *(1) If each user is to have their own unique home-folder then you must create one row for each user. Set the OwnerID to the GUID of the new user. (2) If any of OwnerID, GroupID or Permissions is NULL then the permissions will be inherited from the parent.*

| Field | SQL Type | .NET Type | Required | Default | Comments |
|---|---|---|---|---|---|
| NodeID | uniqueidentifier | Guid | yes | | Generate new for each folder |
| Configuration | nvarchar(4000) | string | yes | | Full Windows path for Windows folder. Append ?%AutoCreate%" to path to have CompleteFTP create the Windows folder when it's first accessed by user. |
| GroupID | uniqueidentifier | Guid | no | | "8a9b1ba4-5d1b-41f6-893a-f99a319be033" for users group. |
| Name | nvarchar(100) | string | yes | | Must be unique amongst nodes with same ParentNodeID |
| NodeTypeID | uniqueidentifier | Guid | yes | | "c3435efd-3b78-44b1-b270-79667906afa9" for Windows folder |
| OwnerID | uniqueidentifier | Guid | no | | GUID of user who owns the folder |
| ParentNodeID | uniqueidentifier | Guid | no | | NULL for root folder, otherwise GUID of parent folder |
| Permissions | nvarchar(12) | string | no | | UNIX-style permissions, e.g. 750 is full-control by owner, read-only by group and none for anyone else. |
| System | bit | bool | yes | | Leave as NULL or set to FALSE. Never set to TRUE. |
| CreatedTime | datetime | DateTime | yes | | Set only when creating |
| ModifiedTime | datetime | DateTime | yes | | Update every time a change is made |

## Table: SiteUser

***Purpose:*** *Stores site-specific information for users - mainly whether or not the user is enabled on the site and what its home-folder is.*

***Notes:*** *You must create one for every site-user combination. By default there are two sites ('Default Site' and 'Admin') so you must create at least two rows in this table for each user, each with a different SiteID (see below).*

| Field | SQL Type | .NET Type | Required | Default | Comments |
|---|---|---|---|---|---|
| SiteID | uniqueidentifier | Guid | yes | | "c7a09c8d-4a7c-4b17-90fd-9897562ee979" for default site and "d7183a00-f941-4b89-a3dc-bd3d1a79c2ca" for admin site. |
| UserID | uniqueidentifier | Guid | yes | | GUID of user |
| Enabled | bit | bool | yes | TRUE | TRUE for default site and FALSE for admin site |
| NodeID | uniqueidentifier | Guid | no | | GUID of home folder. May only be NULL if Enabled is FALSE |
| CreatedTime | datetime | DateTime | yes | | Set only when creating |
| ModifiedTime | datetime | DateTime | yes | | Update every time a change is made |

## Table: Membership

***Purpose:*** *Stores membership of users to groups.*

***Notes:*** *Add at least one row to make the new user a member of the 'users' group.*

| Field | SQL Type | .NET Type | Required | Default | Comments |
|---|---|---|---|---|---|
| UserID | uniqueidentifier | Guid | yes | | GUID of user |
| GroupID | uniqueidentifier | Guid | yes | | "8a9b1ba4-5d1b-41f6-893a-f99a319be033" for users group. |
| CreatedTime | datetime | DateTime | yes | | Set only when creating |
| ModifiedTime | datetime | DateTime | yes | | Update every time a change is made |

## Table: Group

***Purpose:*** *Stores user-groups.*

***Notes:*** *-*

| Field | SQL Type | .NET Type | Required | Default | Comments |
|---|---|---|---|---|---|
| GroupID | uniqueidentifier | Guid | yes | | GUID of group |
| Name | nvarchar(100) | string | yes | | Must be unique |
| System | bit | bool | no | | Leave as NULL or set to FALSE. Never set to TRUE. |
| CreatedTime | datetime | DateTime | yes | | Set only when creating |
| ModifiedTime | datetime | DateTime | yes | | Update every time a change is made |

## Table: Server

***Purpose:*** *Stores information about servers, including currently available IP addresses and activation keys. For stand-alone installations there will be only one row. In clusters each machine will have one row.*

***Notes:*** *There must be one row for each node in a cluster. For stand-alone installations there will only be one row. MachineName, LatestMachineID and IPAddresses are set automatically by the CompleteFTP Service.*

| Field | SQL Type | .NET Type | Required | Default | Comments |
|---|---|---|---|---|---|
| ServerID | uniqueidentifier | Guid | yes | | GUID of the server |
| ActivationKey | nvarchar(1000) | string | no | | Unlocks the purchased edition of CompleteFTP on this machine |

| | | | | | |
|---|---|---|---|---|---|
| ConnectionInfo | ntext | string | no | | Information about how to connect to this server remotely. This field is encrypted and can only be set using CompleteFTP Manager. |
| IPAddresses | nvarchar(1000) | string | no | | Do not set this field. |
| IsPrimary | bit | bool | no | | Is this the primary node in the cluster? |
| LatestMachineID | uniqueidentifier | string | no | | Do not set this field. |
| LogLevel | nvarchar(40) | string | no | | Level of logging produced on this machine. Must be on of: off, error, warning, information, debug or all. |
| MachineName | nvarchar(100) | string | no | | NetBIOS name of machine [Set automatically] |
| Name | nvarchar(100) | string | yes | | Must be unique |
| SyncEnabled | bit | bool | no | | Is synchronization enabled for this cluster-node? |
| SyncPeriod | int | int | no | | Milliseconds between synchronizations |
| CreatedTime | datetime | DateTime | yes | | Set only when creating |
| ModifiedTime | datetime | DateTime | yes | | Update every time a change is made |

## Table: Site

***Purpose:*** *Stores information about sites.*
***Notes:*** *By default there are two sites: one for administration and one for general users. The admin site must always exist, but the other site may be replaced.*

| Field | SQL Type | .NET Type | Required | Default | Comments |
|---|---|---|---|---|---|
| SiteID | uniqueidentifier | Guid | yes | | GUID of the site |
| AnonymousEnabled | bit | bool | | | Are anonymous FTP connections allowed? |
| AnonymousHTTPEnabled | bit | bool | no | TRUE | Is anonymous HTTP enabled on this site? |
| AnonymousMaxConnections | int | int | | | Maximum number of anonymous FTP connections. |
| AnonymousUserID | uniqueidentifier | Guid | | | UserID of the anonymous user. |
| ArchiveNavEnabled | bit | bool | no | FALSE | Is ZIP-file navigation enabled? |
| ArchiveNavSuffix | nvarchar(20) | string | no | .folder | Extension to add to ZIP-file when listing ZIP-folder. |
| AutoBanDuration | int | int | no | 60000 | Period (in ms) for which an auto-ban remain in force. |
| AutoBanTriggerCount | int | int | no | 10 | Number of authentication failures until an IP address is banned. |
| AutoBanTriggerPeriod | int | int | no | 3600000 | Period (in ms) within authentication-failures are counted. |
| BackslashPathSep | bit | bool | no | FALSE | Allow backslashes when resolving paths in the file-system. |
| DefaultDomain | nvarchar(100) | string | no | | Default Windows domain to use when authenticating user-names that don't specify a domain. |
| DefaultHomeDirID | uniqueidentifier | Guid | yes | | NodeID of the folder that's the parent to default user home-folders |
| Enabled | bit | bool | yes | | Is this site enabled? |
| Encoding | nvarchar(40) | string | no | | Character-encoding for all strings in all protocols. NULL, the default, means UTF-8. To get a listing of valid values, search for 'code page identifiers' |
| FTPEnabled | bit | bool | yes | | Is FTP enabled on this site? |
| FTPSEnabled | bit | bool | yes | | Is standard (explicit) FTPS enabled on this site? |
| FTPSImplicitEnabled | bit | bool | yes | | Is legacy (implicit) FTP enabled on this site? |
| FTPSVerifyClient | b | bool | yes | | Require FTPS clients to present their certificates? |
| HideProductVersion | b | bool | no | FALSE | Don't show product and version in any communications (set to true for PCI compliance). |
| HomeDirIsRoot | b | bool | no | FALSE | Should user's home-folders be shown as their root folder? |
| HTMLListingTemplate | ntext | string | no | | Template to use for HTML listings. |
| HTTPEnabled | bit | bool | yes | | Is HTTP enabled on this site? |
| HTTPSEnabled | bit | bool | yes | | Is HTTPS enabled on this site? |
| IPFilterID | uniqueidentifier | Guid | no | | IPFilterID of site's IP filter. |
| ListingFormatID | nvarchar(40) | string | yes | | FTP directory listing format. 'UNIX' for UNIX-style listings, 'WINDOWS' for Windows-style listings or 'FILENAME' for listings that only show file-names. |
| MaxConnections | int | int | no | | Maximum number of simultaneous (non-HTTP) connections allowed for this site |
| MaxConnectionsHTTP | int | int | yes | 200 | Maximum number of simultaneous HTTP connections allowed for this site |

| | | | | | |
|---|---|---|---|---|---|
| MaxLoginAttempts | int | int | no | | Maximum number of login attempts allowed in a single connection |
| Name | nvarchar(100) | string | yes | | Must be unique |
| PASVIP | nvarchar(100) | string | no | | IP address to tell clients to connect to for passive-mode transfers |
| PASVPortMax | int | int | no | | Maximum port-number to use for passive-mode transfers |
| PASVPortMin | int | int | no | | Minimum port-number to use for passive-mode transfers |
| PortFTP | int | int | yes | | Port for FTP and standard (explicit) FTPS |
| PortFTPSImplicit | int | int | yes | | Port for legacy (implicit) FTPS |
| PortHTTP | int | int | yes | | Port for HTTP |
| PortHTTPS | int | int | yes | | Port for HTTPS |
| PortSFTP | int | int | yes | | Port for SFTP, SCP and SSH terminal |
| SCPEnabled | bit | bool | yes | | Is SP enabled on this site? |
| SSHAuthTypes | int | int | no | | SSH authentication-types allowed for SSH (null by default) |
| SSHCipher | int | int | no | | Ciphers allowed for SSH (null by default) |
| SSHCompression | int | int | no | | Compression algorithms allowed for SSH (null by default) |
| SSHKeyAlgorithm | int | int | no | | Key-types allowed for SSH (null by default) |
| SSHKeyExchange | int | int | no | | Key-exchange algorithms allowed for SSH (null by default) |
| SSHKeyPairDSA | image | byte[] | yes | | Site's DSA private key stored in SECSH format. |
| SSHKeyPairRSA | image | byte[] | yes | | Site's RSA private key stored in SECSH format. |
| SSHMAC | int | int | no | | MAC algorithms allowed for SSH (null by default) |
| SSHTerminalEnabled | bit | bool | yes | | Is SSH terminal access enabled on this site? |
| SSLCertificate | image | byte[] | yes | | Site's SSL/TLS certificate in PFX format (by default no password is used) |
| SSLCertificatePassword | nvarchar(100) | string | no | | Encrypted password for certificate (null by default) |
| SSLCipherSuites | int | int | no | | Cipher-suites allowed for SSL/TLS (null by default) |
| System | bit | bool | no | | Is this a system site? Must be FALSE for all sites except the admin site. |
| TimeoutHTTP | int | int | no | 1800000 | Time-out period for HTTP sessions. |
| TimeoutIdle | int | int | no | | Milliseconds of inactivity before a connection is closed. |
| TimeoutLogin | int | int | no | | Milliseconds allowed to log in. |
| TimeoutPassiveWait | int | int | no | | Milliseconds to wait for a passive channel connections before giving up. |
| TimeoutStalled | int | int | no | | Milliseconds of inactivity before a transfer is considered to have stalled. |
| WelcomeMessage | nvarchar(2000) | string | no | | Message shown to users when they connect to this site |
| CreatedTime | datetime | DateTime | yes | | Set only when creating |
| ModifiedTime | datetime | DateTime | yes | | Update every time a change is made |

## Table: ServerSite

***Purpose:*** *Stores server-specific information about sites, including enabled-status and IP bindings. For each server in the configuration there will be a row for each site.*

***Notes:***

| Field | SQL Type | .NET Type | Required | Default | Comments |
|---|---|---|---|---|---|
| ServerID | uniqueidentifier | Guid | yes | | ServerID of the server |
| SiteID | uniqueidentifier | Guid | yes | | SiteID of the site |
| Enabled | bit | bool | no | TRUE | Is the specified site enabled on the specified server? |
| FTPInterfaces | nvarchar(1024) | string | no | | IP addresses to listen on for FTP (null means all). |
| HTTPInterfaces | nvarchar(1024) | string | no | | IP addresses to listen on for HTTP (null means all). |
| SFTPInterfaces | nvarchar(1024) | string | no | | IP addresses to listen on for SFTP (null means all). |
| CreatedTime | datetime | DateTime | yes | | Set only when creating |
| ModifiedTime | datetime | DateTime | yes | | Update every time a change is made |

## Table: NodeType

***Purpose:*** *Defines the various types of folders in the virtual file-system.*

***Notes:*** *Every node-type must have a plug-in of the "File System" type (PlugInTypeID=0).*

| Field | SQL Type | .NET Type | Required | Default | Comments |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| NodeTypeID | uniqueidentifier | Guid | yes | | GUID of the node-type |
| CanHaveChildren | bit | bool | yes | | Can folders of this type have sub-folders? |
| Name | nvarchar(100) | string | yes | | Must be unique |
| PlugInID | uniqueidentifier | Guid | yes | | PlugInID of the plug-in defining the class that implements this node-type. |
| System | bit | bit | no | | Must be FALSE. |
| Visible | bit | bit | no | | Show this folder-type in the list of folder-types when creating new folders. |
| CreatedTime | datetime | DateTime | yes | | Set only when creating |
| ModifiedTime | datetime | DateTime | yes | | Update every time a change is made |

## Table: PlugIn

***Purpose:*** *Defines plug-ins (aka extensions) of all types*

***Notes:*** *Extensions of the "File System" type (PlugInTypeID=0) will not be usable unless a NodeType has been defined for them. Extensions of the "Authentication" type (PlugInTypeID=3) must have a row for each site (allowing distinct configurations for each site).*

| Field | SQL Type | .NET Type | Required | Default | Comments |
|---|---|---|---|---|---|
| PlugInID | uniqueidentifier | Guid | yes | | GUID of the plug-in |
| AssemblyPath | nvarchar(1024) | string | no | | NULL or path to assembly that contains the class |
| ClassName | nvarchar(400) | string | yes | | Full name of class that implements the plug-in. |
| Configuration | ntext | string | no | | Configuration data for the plug-in. Note that FileSystemAdapter plug-ins also have the option of storing configuration information in each individual Node row that references the plug-in. |
| EditorClassName | nvarchar(100) | string | no | | Name of editor class (optional) |
| Name | nvarchar(100) | string | yes | | Must be unique |
| PlugInTypeID | uniqueidentifier | Guid | yes | | PlugInTypeID of the plug-in-type of this plug-in. |
| System | bit | bool | yes | | Must be FALSE. |
| CreatedTime | datetime | DateTime | yes | | Set only when creating |
| ModifiedTime | datetime | DateTime | yes | | Update every time a change is made |

## Table: PlugInType

***Purpose:*** *Enumeration of plug-in types.*

***Notes:*** *Do not modify this table. Values are (0=File System, 1=Custom Commands, 2=Event, 3=Authentication)*

| Field | SQL Type | .NET Type | Required | Default | Comments |
|---|---|---|---|---|---|
| PlugInTypeID | int | int | yes | | See notes |
| Name | nvarchar(100) | string | yes | | See notes |
| CreatedTime | datetime | DateTime | yes | | See notes |
| ModifiedTime | datetime | DateTime | yes | | See notes |

## Table: Authenticator

***Purpose:*** *Stores authenticator information for specific sites.*

***Notes:*** *Plug-ins must be of the Authentication type (i.e. PlugInTypeID=3). There must be one row for each site-authenticator combination).*

| Field | SQL Type | .NET Type | Required | Default | Comments |
|---|---|---|---|---|---|
| PlugInID | uniqueidentifier | Guid | yes | | PlugInID of the plug-in. |
| SiteID | uniqueidentifier | Guid | yes | | SiteID of the site. |
| UserID | uniqueidentifier | Guid | yes | | UserID of the log-in-as user that is used as a template for users logging in using the references authenticator. |
| Configuration | nvarchar(4000) | string | no | | Site-specific authentication configuration. |
| Enabled | bit | bool | yes | | Is this authenticator enabled on this site? |
| CreatedTime | datetime | DateTime | yes | | Set only when creating |
| ModifiedTime | datetime | DateTime | yes | | Update every time a change is made |

# How to set up a secure server

CompleteFTP supports secure FTP via the SFTP protocol (FTP over SSH), and the FTPS protocol (or FTP over SSL). CompleteFTP also supports SCP and HTTPS (but not in the *Standard Edition*).

**FTPS**

FTPS may be enabled by selecting the "FTPS enabled" setting (see the FTP/FTPS group of settings), and its use can be made mandatory (i.e. no 'plain' insecure connections) by unselecting the "FTP enabled" setting. In this case users can only use FTPS, and ordinary FTP connections will be rejected (i.e. initial connection to the server must be followed by the AUTH command).

By default, the FTPS and FTP port is port 21.

**SFTP**

SFTP may be enabled by selecting the "SFTP enabled" setting (see the SFTP/SCP group of settings). Similarly, SCP may be enabled by selecting the "SCP enabled" setting.

By default, the SFTP and SCP port is port 22.

SFTP, SCP and FTPS can be enabled simultaneously (and by default are).

**HTTPS**

HTTPS (which is HTTP secured over SSL/TLS) may be enabled by selecting the "HTTPS enabled" setting (see the HTTP/HTTPS group of settings).

# How to use HTTP

CompleteFTP supports both public HTTP access to files, and authenticated access to files. HTTP directory listings are also supported (if listing permissions are in place).

**Public access**

For public (unauthenticated) access, the "anonymous" user (see Users Panel) must be enabled and the "Public HTTP access enabled" setting must be checked.

Only the files accessible by the anonymous user can be viewed publicly via HTTP (i.e. by a web browser without logging in). When the server is being accessed publicly the root of file-system is always the anonymous user's home directory, thus http://hostname/index.html maps to /Public/index.html in the virtual file-system, assuming that /Public is the anonymous user's home directory (the default).

Any access to a path that is not accessible to the anonymous user will result in an authentication request (i.e. login dialog in the browser). The user will thus be given a chance to authenticate. Alternatively, the default index.html file has a *Login* link that can be used to login (or this link, */login*, can be placed on any page desired).

**Private access**

A particular user's files can be accessed via HTTP if the user is authenticated. For example, the user 'javaftp' normally has their files accessible under '/home/javaftp' (as long as the General User Setting *User's home folder appears as root* is not selected). So all of the 'javaftp' user's files will be accessible via 'http://hostname/home/javaftp/filepath'. When an attempt is made to navigate to this URL in a web browser, an authentication dialog will be popped up by the browser, and the 'javaftp' user's credentials can be entered and a session established. Note that in HTTP all data sent to the server is not secure, especially usernames and passwords.

If *User's home folder appears as root* is enabled, the authenticated user's files are accessed ia 'http://hostname/filepath', i.e. the '/home/javaftp' part of the URL is not required. The login url http://hostname/login can be used to authenticate if necessary.

**Sessions**

HTTP sessions are maintained by a cookie stored in the browser. The timeout for HTTP sessions can be set in Limits & Timeouts (in Settings). A session can also be terminated by navigating to http://hostname/logout, and can be initiated by navigating to http://hostname/login.

# How to use HTTPS

The HTTPS protocol is the HTTP protocol secured by an SSL/TLS connection. It should be used for when usernames and passwords (as well as any other sensitive data) is sent via HTTP.

A server certificate is required for HTTPS, and one is generated on installation and can be modified via Settings->FTP/FTPS->Advanced FTP/FTPS Settings->Security Settings. The default certificate does not have your hostname, and so browsers will raise a warning when the site is navigated to using HTTPS. A certificate authority (CA) must be contacted to purchase a certificate for your domain to remove the browser warning. To do this, a Certificate Signing Request (CSR) must be sent to the CA. See how to generate a CSR for more details.

# How to perform HTTP uploads

CompleteFTP supports HTTP file uploads to directories where uploads are permitted by the authenticated user.

This uses the standard *input* element with *type="file"* in an HTML form. For example:

```
<FORM METHOD=POST ENCTYPE="multipart/form-data" ACTION=".">
  <INPUT TYPE=FILE NAME="upfile"><INPUT TYPE=SUBMIT VALUE="Upload File">
</FORM>
```

If a directory is writable, the CompleteFTP directory listing page will include an HTML form permitting HTTP file uploads into this directory.

## Multiple sites

The *Enterprise Edition* of CompleteFTP supports the creation and administration of multiple sites, in addition to the "default site" (consisting of the default ports for FTP,SFTP and HTTP/HTTPS) and the "admin site" (the port connected to by the CompleteFTP manager to administer CompleteFTP, 14982 (for FTPS) and 14983 (for SFTP) by default.

For example, it may be desirable to have internal users use the default user site, and to create a completely different site, using different port numbers, for users external to the organisation. This could be called the "external user site". The external user port for FTP might be 10021, and the external user port for SFTP might be 10022.

In this situation the external firewall would be configured to forward external connections to the correct internal port numbers, which would be different to the standard port numbers for FTP and SFTP. So external FTP users connecting to port 21 on the firewall would be forwarded to port 10021, the port on

Users can also be restricted to use certain sites, so that a separate group of logins can be maintained for external users.

In the *Enterprise Edition*, the available sites are listed on the left hand side of the *Sites* panel. Sites can be added via the 'Add site' link, and deleted via the 'Remove' link. Extreme care should be taken when deleting sites.

The Admin site is displayed in *Sites* by unchecking the 'Show admin site' checkbox at the bottom left of the *Sites* panel. The Admin site cannot be deleted, as it is required to administer CompleteFTP.

## NAME

**mustache** -- Logic-less templates.

## SYNOPSIS

A typical Mustache template:

```
Hello {{name}}
You have just won ${{value}}!
{{#in_ca}}
Well, ${{taxed_value}}, after taxes.
{{/in_ca}}
```

Given the following hash:

```
{
  "name": "Chris",
  "value": 10000,
  "taxed_value": 10000 - (10000 * 0.4),
  "in_ca": true
}
```

Will produce the following:

```
Hello Chris
You have just won $10000!
Well, $6000.0, after taxes.
```

## DESCRIPTION

Mustache can be used for HTML, config files, source code - anything. It works by expanding tags in a template using values provided in a hash or object.

We call it "logic-less" because there are no if statements, else clauses, or for loops. Instead there are only tags. Some tags are replaced with a value, some nothing, and others a series of values. This document explains the different types of Mustache tags.

## TAG TYPES

Tags are indicated by the double mustaches. **{{person}}** is a tag, as is **{{#person}}**. In both examples, we'd refer to **person** as the key or tag key. Let's talk about the different types of tags.

### Variables

The most basic tag type is the variable. A **{{name}}** tag in a basic template will try to find the **name** key in the current context. If there is no **name** key, nothing will be rendered.

All variables are HTML escaped by default. If you want to return unescaped HTML, use the triple mustache: **{{{name}}}**.

You can also use **&** to unescape a variable: **{{& name}}**. This may be useful when changing delimiters (see "Set Delimiter" below).

By default a variable "miss" returns an empty string. This can usually be configured in your Mustache library. The Ruby version of Mustache supports raising an exception in this situation, for instance.

Template:

```
* {{name}}
* {{age}}
* {{company}}
* {{{company}}}
```

Hash:

```
{
  "name": "Chris",
  "company": "<b>GitHub</b>"
}
```

Output:

```
* Chris
*
* &lt;b&gt;GitHub&lt;/b&gt;
* <b>GitHub</b>
```

**Sections**

Sections render blocks of text one or more times, depending on the value of the key in the current context.

A section begins with a pound and ends with a slash. That is, **{{#person}}** begins a "person" section while **{{/person}}** ends it.

The behavior of the section is determined by the value of the key.

**False Values or Empty Lists**

If the **person** key exists and has a value of false or an empty list, the HTML between the pound and slash will not be displayed.

Template:

```
Shown.
{{#nothin}}
  Never shown!
{{/nothin}}
```

Hash:

```
{
  "person": true,
}
```

Output:

```
Shown.
```

**Non-Empty Lists**

If the **person** key exists and has a non-false value, the HTML between the pound and slash will be rendered and displayed one or more times.

When the value is a non-empty list, the text in the block will be displayed once for each item in the list. The context of the block will be set to the current item for each iteration. In this way we can loop over collections.

Template:

```
{{#repo}}
  <b>{{name}}</b>
{{/repo}}
```

Hash:

```
{
  "repo": [
    { "name": "resque" },
    { "name": "hub" },
    { "name": "rip" },
  ]
}
```

Output:

```
<b>resque</b>
<b>hub</b>
<b>rip</b>
```

**Lambdas**

When the value is a callable object, such as a function or lambda, the object will be invoked and passed the block of text. The text passed is the literal block, unrendered. **{{tags}}** will not have been expanded - the lambda should do that on its own. In this way you can implement filters or caching.

Template:

```
{{#wrapped}}
  {{name}} is awesome.
{{/wrapped}}
```

Hash:

```
{
  "name": "Willy",
  "wrapped": function() {
    return function(text) {
      return "<b>" + render(text) + "</b>"
```

```
    }
  }
}
```

Output:

```
<b>Willy is awesome.</b>
```

**Non-False Values**

When the value is non-false but not a list, it will be used as the context for a single rendering of the block.

Template:

```
{{#person?}}
  Hi {{name}}!
{{/person?}}
```

Hash:

```
{
  "person?": { "name": "Jon" }
}
```

Output:

```
Hi Jon!
```

**Inverted Sections**

An inverted section begins with a caret (hat) and ends with a slash. That is **{{^person}}** begins a "person" inverted section while **{{/person}}** ends it.

While sections can be used to render text one or more times based on the value of the key, inverted sections may render text once based on the inverse value of the key. That is, they will be rendered if the key doesn't exist, is false, or is an empty list.

Template:

```
{{#repo}}
  <b>{{name}}</b>
{{/repo}}
{{^repo}}
  No repos :(
{{/repo}}
```

Hash:

```
{
  "repo": []
}
```

Output:

```
No repos :(
```

**Comments**

Comments begin with a bang and are ignored. The following template:

```
<h1>Today{{! ignore me }}.</h1>
```

Will render as follows:

```
<h1>Today.</h1>
```

Comments may contain newlines.

**Partials**

Partials begin with a greater than sign, like **{{> box}}**.

Partials are rendered at runtime (as opposed to compile time), so recursive partials are possible. Just avoid infinite loops.

They also inherit the calling context. Whereas in ERB you may have this:

```
<%= partial :next_more, :start => start, :size => size %>
```

Mustache requires only this:

```
{{> next_more}}
```

Why? Because the **next_more.mustache** file will inherit the **size** and **start** methods from the calling context.

In this way you may want to think of partials as includes, or template expansion, even though it's not literally true.

For example, this template and partial:

```
base.mustache:
<h2>Names</h2>
{{#names}}
  {{> user}}
{{/names}}

user.mustache:
<strong>{{name}}</strong>
```

Can be thought of as a single, expanded template:

```
<h2>Names</h2>
{{#names}}
  <strong>{{name}}</strong>
{{/names}}
```

**Set Delimiter**

Set Delimiter tags start with an equal sign and change the tag delimiters from **{{** and **}}** to custom strings.

Consider the following contrived example:

```
* {{default_tags}}
{{=<% %>=}}
* <% erb_style_tags %>
<%={{ }}=%>
* {{ default_tags_again }}
```

Here we have a list with three items. The first item uses the default tag style, the second uses erb style as defined by the Set Delimiter tag, and the third returns to the default style after yet another Set Delimiter declaration.

According to ctemplates, this "is useful for languages like TeX, where double-braces may occur in the text and are awkward to use for markup."

Custom delimiters may not contain whitespace or the equals sign.

## COPYRIGHT

Mustache is Copyright (C) 2009 Chris Wanstrath

Original CTemplate by Google

## SEE ALSO

mustache(1), mustache(7), http://mustache.github.com/

# How to activate an installation of CompleteFTP

When a production license has been purchased, the trial installation must be *activated*.

This usually involves accessing the *CompleteFTP Activation System* on the Internet, but from 7.1.2, customers who have purchased Corporate licenses can request a universal activation key which does not require Internet access.

The following steps should be followed if you do **not** have a universal AK, which will be the majority of users.

A. Run the CompleteFTP manager and connect to the CompleteFTP server instance that you wish to activate (the server instance - the CompleteFTP Windows service - may be running on another machine). You should run the manager on a machine with internet access (although there is a work around) if you do not have a universal activation key.
B. Select the "Licensing" menu item from the list on the left in the manager, and then select the "Activate your license" link.
C. Enter your purchase reference in the edit box indicated (Step 1), and then open the supplied URL. You will need the login details provided in your purchase email. If the credentials and purchase reference are correct, you will see a web page called "CompleteFTP Activation System".
D. Your activation key should be displayed in the text box, and it can be copied and pasted into the CompleteFTP manager in Step 4. Select "OK" and it should now be activated, and the license details should be displayed.

The following steps should be followed if you **do** have a universal AK:

A. Run the CompleteFTP manager and connect to the CompleteFTP server instance that you wish to activate (the server instance - the CompleteFTP Windows service - may be running on another machine).
B. Select the "Licensing" menu item from the list on the left in the manager, and then select the "Activate your license" link.
C. Paste the universal AK you have been sent into the CompleteFTP manager in Step 4. Select "OK" and it should now be activated, and the license details should be displayed.

If you need to move an instance of CompleteFTP from one server to another, or are having difficulties with activating, please email us.

## Universal activation keys

CompleteFTP Corporate licenses entitles users to a *universal activation key* (UAK) for activating CompleteFTP installations.

Normally, activation keys are tied to a specific Windows installation via the Windows SID (Security Identifier). Installations need to be activated via the Internet (although the specific machine being activated does not require Internet access - the activation key can be obtained using a machine with Internet access).

A UAK is not tied to the Windows SID, so the same activation key will work on all installations. This means it can be hard-coded into silent installations. The UAK has the name of the licensed owner encrypted in it, which is displayed in the CompleteFTP manager's title bar.